**Important Information**

Texas Instruments makes no warranty, either express or implied, including but not limited to any implied warranties of merchantability and fitness for a particular purpose, regarding any programs or book materials and makes such materials available solely on an "as-is" basis. In no event shall Texas Instruments be liable to anyone for special, collateral, incidental, or consequential damages in connection with or arising out of the purchase or use of these materials, and the sole and exclusive liability of Texas Instruments, regardless of the form of action, shall not exceed the purchase price of this product. Moreover, Texas Instruments shall not be liable for any claim of any kind whatsoever against the use of these materials by any other party.

**USA FCC Information Concerning Radio Frequency Interference**

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, you can try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
• Consult the dealer or an experienced radio/television technician for help.

Caution: Any changes or modifications to this equipment not expressly approved by Texas Instruments may void your authority to operate the equipment.

© 2005 Texas Instruments Incorporated
Windows and Macintosh are trademarks of their respective owners.
Installing the AAA Batteries

The TI-89 Titanium uses four AAA alkaline batteries and a backup silver oxide battery (SR44SW or 303). The backup battery is already installed, and the AAA batteries are provided with the product.

1. Remove the battery cover from the back of the calculator.

2. Unwrap the four AAA batteries provided with your product and insert them in the battery compartment. Arrange the batteries according to the polarity (+ and −) diagram in the battery compartment.

3. Replace the battery cover on the calculator. The cover should snap into place.
Turning on your TI-89 Titanium for the first time

After installing the batteries included with the calculator, press ON. The Apps desktop appears.

**Note:** If your calculator initializes the preinstalled Apps, a progress bar will appear with the message “Installation in progress . . . Do not interrupt!” instead of the Apps desktop. To avoid losing Apps, do not remove the batteries during initialization. (You can re-install Apps from either the Product CD-ROM or education.ti.com.)

Adjusting the contrast

- To lighten the display, press and hold □ and tap −.
- To darken the display, press and hold □ and tap +.

The Apps desktop

The Apps desktop is the starting point for operating your TI-89 Titanium. Your installed Apps appear on the Apps desktop as icons organized in categories for easy access. From the Apps desktop, you can:

- Open Apps.
• Select and edit categories of Apps.
• View all of the Apps installed on your calculator.
• View the full name of the highlighted App.
• View and edit the time and date.
• Check status line information.
• View split-screen mode information.

**TI-89 Titanium Apps desktop**

1. View full name of highlighted App.
2. View time and date.
3. Press **ENTER** to open highlighted App.
4. Scroll down to view additional Apps.
5. Check status line information.
Edit categories.

To return to the Apps desktop at any time, press [APPS]. The last category selected appears with the last open App highlighted.

**Turning off the calculator**

Press [2nd] [OFF]. The next time you turn on the calculator, the Apps desktop appears with the same settings and memory contents retained. (If you turned off the Apps desktop, the calculator Home screen appears.)

You can use either of the following keys to turn off the TI-89 Titanium.

<table>
<thead>
<tr>
<th>Press:</th>
<th>Description</th>
</tr>
</thead>
</table>
| [2nd] [OFF] (press [2nd] and then press [OFF]) | Settings and memory contents are retained by the Constant Memory™ feature.  
  • You cannot, however, use [2nd] [OFF] if an error message is displayed.  
  • When you turn the TI-89 Titanium on again, it displays either the Home screen or the Apps desktop (regardless of the last application you used). |
| [◆] [OFF] (press [◆] and then press [OFF]) | Similar to [2nd] [OFF] except:  
  • You can use [◆] [OFF] if an error message is displayed.  
  • When you turn the TI-89 Titanium on again, it will be exactly as you left it. |
Note: [OFF] is the second function of the [ON] key.

The calculator’s Automatic Power Down™ (APD™) feature prolongs battery life by turning the calculator off automatically following several minutes of inactivity. When you turn on the calculator after APD:

- The display, cursor, and any error conditions are exactly the same as before APD.
- All settings and memory contents are retained.

Note: APD does not occur if a calculation or program is in progress, unless the program is paused. If a program is running but waiting for a key press, APD will occur after several minutes of inactivity.
TI-89 Titanium keys

Diagram of the TI-89 Titanium keys with labeled parts:

1. [Label for part 1]
2. [Label for part 2]
3. [Label for part 3]
4. [Label for part 4]
**TI-89 Titanium keys**

1. Function keys (F1–F8) open toolbar menus, access Apps, and edit categories of Apps.

2. Cursor keys (☐, ☚, ☘, ☙) move the cursor.

3. Numeric keypad performs math and scientific functions.

4. Modifier keys (2nd, ☣, ☤) add features by increasing the number of key commands.

**Entering special characters**

Use the CHAR (Character) menu and key commands to enter special characters. The CHAR menu lets you access Greek, math, international, and other special characters. An on-screen keyboard map shows the locations of shortcuts used to enter other commonly used characters.

To select characters from the CHAR menu:

1. Press 2nd [CHAR]. The CHAR menu appears.
2. Use the cursor keys to select a category. A submenu lists the characters in that category.
3. Use the cursor keys to select a character, and press ENTER.
**Example:** Enter the right arrow symbol (→) in the Text Editor.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd [CHAR]</td>
<td>![CHAR menu]</td>
</tr>
<tr>
<td>4</td>
<td>![Special menu]</td>
</tr>
</tbody>
</table>

Scroll down for more characters.

- or -

Press ◀ repeatedly to select 9:→ and press [ENTER]

Symbol displayed at cursor location.

To open the keyboard map, press [KEY]. The keyboard map appears.
To type most characters, press ♠ and the corresponding key. Press [ESC] to close the map.

**Example:** Use the keyboard map to find the “not equal to” symbol (≠) shortcut and enter the symbol in the Program Editor.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>♠ [KEY]</td>
<td><img src="image" alt="Keyboard Map" /></td>
</tr>
<tr>
<td>♠ ≠</td>
<td><img src="image" alt="Symbol Displayed" /></td>
</tr>
</tbody>
</table>
Modifier keys add features by increasing the number of keyboard operations at your fingertips. To access a modifier function, press a modifier key and then press the key for the corresponding operation.

<table>
<thead>
<tr>
<th>Keys</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd</td>
<td>Accesses Apps, menu options, and other operations. Second functions are printed above their corresponding keys in the same color as the 2nd key.</td>
</tr>
<tr>
<td>(Second)</td>
<td></td>
</tr>
<tr>
<td>⬜️</td>
<td>Accesses Apps, menu options, and other operations. Diamond functions are printed above their corresponding keys in the same color as the ⬜️ key.</td>
</tr>
<tr>
<td>(Diamond)</td>
<td></td>
</tr>
<tr>
<td>⬆️</td>
<td>Types an uppercase character for the next letter key you press. Also used with ⬅️ and ⬆️ to highlight characters when editing.</td>
</tr>
<tr>
<td>(Shift)</td>
<td></td>
</tr>
<tr>
<td>alpha</td>
<td>Lets you type alphabetic characters without a QWERTY keypad. Alpha characters are printed above their corresponding keys in the same color as the alpha key.</td>
</tr>
<tr>
<td>(Alpha)</td>
<td></td>
</tr>
</tbody>
</table>
Example: Access the VAR-LINK [All] screen, where you can manage variables and Apps.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd [VAR-LINK]</td>
<td>![VAR-LINK Screen]</td>
</tr>
</tbody>
</table>

**Function keys**

Use the function keys to perform the following operations:

- On the Apps desktop, open Apps and select or edit Apps categories.
- On the calculator Home screen, open toolbar menus to select math-related operations.
- Within Apps, open toolbar menus to select App options.

**Numeric keypad**

The numeric keypad lets you enter positive and negative numbers.

To enter a negative number, press \( (-) \) before typing the number.

**Note:** Don’t confuse the negation key (\( (-) \)) with the subtraction key (\( (-) \)).
To enter a number in scientific notation:

1. Type the numbers that precede the exponent. (This value can be an expression.)
2. Press EE. The exponent symbol (E) follows the numbers you entered.
3. Type the exponent as an integer with up to three digits. (As the following example shows, you can use a negative exponent.)

**Example:** On the calculator Home screen, enter 0.00685 using scientific notation.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 8 5</td>
<td><img src="image1.png" alt="Calculator Screen" /></td>
</tr>
<tr>
<td>EE</td>
<td><img src="image2.png" alt="Calculator Screen" /></td>
</tr>
<tr>
<td>(-) 3</td>
<td><img src="image3.png" alt="Calculator Screen" /></td>
</tr>
<tr>
<td>ENTER</td>
<td><img src="image4.png" alt="Calculator Screen" /></td>
</tr>
</tbody>
</table>

**Other important keys**

<table>
<thead>
<tr>
<th>Key Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Y=]</td>
<td>Displays the Y= Editor.</td>
</tr>
<tr>
<td>Key Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>[WINDOW]</td>
<td>Displays the Window Editor.</td>
</tr>
<tr>
<td>[GRAPH]</td>
<td>Displays the Graph screen.</td>
</tr>
<tr>
<td>[TBLSET]</td>
<td>Sets parameters for the Table screen.</td>
</tr>
<tr>
<td>[TABLE]</td>
<td>Displays the Table screen.</td>
</tr>
<tr>
<td>[CUT]</td>
<td>These keys let you edit entered information by performing a cut, copy, or paste operation.</td>
</tr>
<tr>
<td>[COPY]</td>
<td></td>
</tr>
<tr>
<td>[PASTE]</td>
<td></td>
</tr>
<tr>
<td>[APP]</td>
<td>Displays the Apps desktop.</td>
</tr>
<tr>
<td>[_APPS]</td>
<td>With the Apps desktop off, displays the FLASH APPLICATIONS menu.</td>
</tr>
<tr>
<td>2nd [++]</td>
<td>Switches between the last two chosen Apps.</td>
</tr>
<tr>
<td>2nd [CUSTOM]</td>
<td>Turns the custom menu on and off.</td>
</tr>
<tr>
<td>2nd [▶]</td>
<td>Converts measurement units.</td>
</tr>
<tr>
<td>[–]</td>
<td>Designates a measurement unit.</td>
</tr>
<tr>
<td>[←]</td>
<td>Deletes the character to the left of the cursor (backspace).</td>
</tr>
<tr>
<td>[DEL]</td>
<td>Deletes the character to the right of the cursor.</td>
</tr>
<tr>
<td>2nd [INS]</td>
<td>Switches between insert and overwrite modes.</td>
</tr>
</tbody>
</table>
Mode settings

Modes control how the TI-89 Titanium displays and interprets information. All numbers, including elements of matrices and lists, are displayed according to the current mode settings. When the TI-89 Titanium is turned off, the Constant Memory™ feature retains all of the mode settings you have selected.

To view the TI-89 Titanium mode settings:

<table>
<thead>
<tr>
<th>Key Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd [MEM]</td>
<td>Displays the MEMORY screen.</td>
</tr>
<tr>
<td>CATALOG</td>
<td>Displays a list of commands.</td>
</tr>
<tr>
<td>2nd [RCL]</td>
<td>Recalls the contents of a variable.</td>
</tr>
<tr>
<td>STO►</td>
<td>Stores a value to a variable.</td>
</tr>
<tr>
<td>2nd [CHAR]</td>
<td>Displays the CHAR menu, which lets you select Greek letters, international accented characters, and other special characters.</td>
</tr>
</tbody>
</table>
| 2nd [QUIT]  | • In full-screen mode, displays the Apps desktop.  

• In split-screen mode, displays the full-screen view of the active App.  

• With the Apps desktop off, displays the calculator Home screen. |
1. Press `MODE`. Page 1 of the MODE dialog box appears.

2. Press `F2` or `F3` to display the modes listed on Page 2 or Page 3.

**Note:** Modes that are grayed out are available only if other required mode settings are selected. For example, the Custom Units mode listed on Page 3 is available only if the Unit System mode is set to CUSTOM.

### Viewing mode settings

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MODE</code></td>
<td><img src="image" alt="Mode settings" /></td>
</tr>
<tr>
<td><code>F2</code></td>
<td><img src="image" alt="Mode settings" /></td>
</tr>
</tbody>
</table>
Changing mode settings

**Example:** Change the Language mode setting to Spanish (Español).

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>F3</strong></td>
<td><img src="image" alt="MODE setting" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MODE</strong></td>
<td><img src="image" alt="MODE setting" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>F3</strong></td>
<td><img src="image" alt="MODE setting" /></td>
</tr>
</tbody>
</table>
### Getting Started

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scroll down to the Language field. &lt;br&gt;👇</td>
<td>![Mode menu with Language list highlighted to 3:Espanol]</td>
</tr>
<tr>
<td>Press ↑ &lt;br&gt;and then press ◀ until 3:Espanol is highlighted.</td>
<td>![Mode menu with Language list and 3:Espanol highlighted]</td>
</tr>
</tbody>
</table>

**Note:** Your menu list might vary, depending on the languages installed.

**Press ENTER**

![Mode menu with Language list and 3:Espanol highlighted]
To return the Language mode setting to English, repeat the steps, selecting 1:English in the Language field.

**Using the Catalog to access commands**

Use the Catalog to access a list of TI-89 Titanium commands, including functions, instructions, and user-defined programs. Commands are listed alphabetically. Commands not beginning with a letter are found at the end of the list (&, /, +, –, etc.).

The Catalog Help App includes details about each command.

Options not currently valid are grayed out. For example, the Flash Apps (F3) menu option is grayed out if no Flash applications are installed on your TI-89 Titanium; the User-Defined (F4) menu option is grayed out if you have not created a function or program.
**Note:** Typing a letter takes you to the first command in the list starting with the same letter.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CATALOG</strong>&lt;br&gt;(displays Built-in commands)</td>
<td><img src="image1.png" alt="Built-in commands screen" /></td>
</tr>
<tr>
<td><strong>F3</strong>&lt;br&gt;(displays Flash Apps commands, if any)</td>
<td><img src="image2.png" alt="Flash Apps commands screen" /></td>
</tr>
<tr>
<td><strong>F4</strong>&lt;br&gt;(displays User-Defined commands, if any)</td>
<td><img src="image3.png" alt="User-Defined commands screen" /></td>
</tr>
</tbody>
</table>

Select commands from the Catalog and insert them onto the calculator Home screen entry line or paste them to other Apps, such as the Y= Editor, Text Editor, or CellSheet™ Apps.
**Example:** Insert the `comDenom` command on the calculator Home screen entry line.

**Note:** Before selecting a command, position the cursor where you want the command to appear.

Pressing `2nd` ↘ advances the Catalog list one page at a time.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOG</td>
<td><img src="image" alt="Catalog List" /></td>
</tr>
<tr>
<td>2nd ↘</td>
<td><img src="image" alt="Selected Command" /></td>
</tr>
</tbody>
</table>

Then press ↘ until the pointer is at the `comDenom` function.

| ENTER       | ![Selected Command](image) |

The status line displays any required and optional parameters for the selected command. Optional parameters appear in square brackets.

**Note:** Pressing `F1` will also display the parameters for the selected command.
To exit the Catalog without selecting a command, press **ESC**.

**Calculator Home screen**

The calculator Home screen is the starting point for math operations, including executing instructions, evaluating expressions, and viewing results.

To display the calculator Home screen, press: **HOME**

You can also display the calculator Home screen from the Apps desktop by highlighting the Home icon and pressing **ENTER**.
1 History area lists the entry/answer pairs entered.

2 Tabs display menus for selecting lists of operations. Press F1, F2, and so on to display menus.

3 Result of last entry is displayed here. (Note that results are not displayed on the entry line.)

4 Status line shows the current state of the calculator.

5 Entry line displays your current entry.

6 Your previous entry is displayed here.

To return to the Apps desktop from the calculator Home screen, press [APPS].
About the history area

The history area displays up to eight entry/answer pairs, depending on the complexity and height of the expressions. When the display is filled, information scrolls off the top of the screen. Use the history area to:

- Review previous entries and answers. Use the cursor keys to view entries and answers that have scrolled off the screen.
- Recall or auto-paste a previous entry or answer onto the entry line to reuse or edit. (For more information, see the electronic Operating the Calculator chapter.)

The cursor, which normally rests on the entry line, can be moved into the history area. The following table shows you how to move the cursor around in the history area.

<table>
<thead>
<tr>
<th>To</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>View entries/answers scrolled off the screen</td>
<td>From the entry line, press [↑] to highlight the last answer. Continue using [↑] to move the cursor from answer to entry through the history area.</td>
</tr>
<tr>
<td>Go to the oldest or newest entry/answer pair</td>
<td>If the cursor is in the history area, press [↑] [←] or [↓] [→].</td>
</tr>
<tr>
<td>View an entry or answer too long for one line ([DEL] is displayed at the end of the line)</td>
<td>Move the cursor to the entry or answer. Use [↓] or [↑] to scroll left or right and [2nd] [←] or [2nd] [→] to go to the beginning or end.</td>
</tr>
<tr>
<td>Return cursor to the entry line</td>
<td>Press [ESC], or press [↑] until the cursor is back on the entry line.</td>
</tr>
</tbody>
</table>
Interpreting history information on the status line

Use the history indicator on the status line for information about the entry/answer pairs. For example:

If the cursor is on the entry line:

Total number of pairs currently saved  8/30  Maximum number of pairs that can be saved

If the cursor is in the history area:

Pair number of the highlighted entry/answer  8/30  Total number of pairs currently saved

Modifying the history area

To change the number of pairs that can be saved:

1. From the calculator Home screen, press F1 and select 9:Format.
2. Press hp and use ↓ or ↑ to highlight the new number.
3. Press ENTER ENTER.

To clear the history area and delete all saved pairs:

• From the calculator Home screen, press F1 and select 8:Clear Home.
  – or –
Enter **ClrHome** on the calculator Home screen entry line.

To delete an entry/answer pair, move the cursor to either the entry or answer, and press ← or CLEAR.

**Working with Apps**

The TI-89 Titanium organizes Apps by category on the Apps desktop. To select a category, press a function key (F2 through 2nd [F8]). The App icons for the selected category appear on the Apps desktop.

**Note:** If the name under an Apps desktop icon is truncated, use the cursor keys to highlight the icon. Now view the full name at the top of the Apps desktop.

**Opening Apps**

Use the cursor keys or press the first letter of the App name to highlight the Apps icon on the Apps desktop and press ENTER. The App either opens directly or displays a dialog box. The most common dialog box lists these options for the App:

**Note:** The TI-89 Titanium uses the general term *variable* to refer to the App data files that you create.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>Returns the screen displayed when you last viewed the App. If no current App variable exists, the New dialog box appears.</td>
</tr>
<tr>
<td>Open</td>
<td>Lets you open an existing file.</td>
</tr>
</tbody>
</table>
Select an option, enter any required information, and press \textbf{ENTER}. The App appears.

\textbf{Example}: Create a new program using the Program Editor.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>Creates a new file with the name typed in the field.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use cursor keys to highlight</td>
<td><img src="image.png" alt="Image" /></td>
</tr>
<tr>
<td>\texttt{F1}</td>
<td><img src="image.png" alt="Image" /></td>
</tr>
<tr>
<td>ENTER</td>
<td><img src="image.png" alt="Image" /></td>
</tr>
<tr>
<td>3</td>
<td><img src="image.png" alt="Image" /></td>
</tr>
</tbody>
</table>
The newly created program variable, *program1*, is saved to the Main folder.
Returning to the Apps desktop from within an App

Press [APPS]. The icons for the last Apps category selected appear on the Apps desktop with the icon for the last App opened highlighted.

You can also return to the Apps desktop by pressing [2nd] [QUIT] in full-screen mode. In split-screen mode, press [2nd] [QUIT] twice.

To return to the last open App from the Apps desktop, press [2nd] [←].

Selecting an Apps category

On the TI-89 Titanium, the Apps category names appear only in the F1 Menu. To select an Apps category, press [F1] 2:Select Category and use the cursor keys to highlight an Apps category, and then press [ENTER] to select the highlighted category. You can also use the function key shortcuts to select a category from the keypad (use the [2nd] key if necessary). The App icons for the selected category appear on the Apps desktop.

The App icons for the selected category appear on the Apps desktop.

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2</td>
<td>All</td>
</tr>
<tr>
<td>F3</td>
<td>English</td>
</tr>
<tr>
<td>F4</td>
<td>SocialSt</td>
</tr>
<tr>
<td>F5</td>
<td>Math</td>
</tr>
</tbody>
</table>
Example: Select the All category.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2</td>
<td>![Image of F2 menu]</td>
</tr>
</tbody>
</table>

If you select an Apps category containing no Apps, a message appears to confirm that the category is empty and point you to the **F1 1:Edit Categories** menu, where you can add App shortcuts to the category. (See “Customizing the Apps categories” on page 33.)

Press **ENTER** or **ESC** to clear the message and return to the Apps desktop.
Customizing the Apps categories

The TI-89 Titanium organizes your Apps into seven categories, six of which you can customize to fit your individual needs. (The All category contains every installed App and cannot be edited.)

To customize the $\text{F3}$ through $\text{2nd F8}$ Apps categories:

1. Select $\text{F1 1:Edit Categories}$. A submenu displays the six customizable Apps category names. (The All category is not listed.)

2. Highlight an Apps category and press $\text{ENTER}$. The Edit Categories dialog box appears with a list of installed Apps and a text box with the category name highlighted.

3. To change the Apps category name, type the desired name.

   **Note:** Enter a name of up to eight characters, including letters with or without capitalization, numbers, punctuation, and accented characters.

4. To add or remove an App shortcut from the category, press $\text{ as required to highlight the box next to the App, then press }$ to add or remove the check mark ($\checkmark$).

5. To save the changes and return to the Apps desktop, press $\text{ENTER}$.
**Example:** Replace the Social Studies category with the Business category and add the CellSheet™ and Finance App shortcuts.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="F1" /></td>
<td><img src="image" alt="Image" /></td>
</tr>
<tr>
<td><img src="image" alt="Image" /></td>
<td><img src="image" alt="Image" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image" alt="Image" /></td>
</tr>
<tr>
<td>or</td>
<td><img src="image" alt="Image" /></td>
</tr>
</tbody>
</table>

---

**Getting Started**

34
<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2nd  [a-lock]</strong></td>
<td>![Image of Edit Categories screen with Business selected]</td>
</tr>
<tr>
<td><strong>B u s i n e s s</strong></td>
<td>![Image of Edit Categories screen with CellSheet selected]</td>
</tr>
<tr>
<td></td>
<td>![Image of Edit Categories screen with Finance selected]</td>
</tr>
</tbody>
</table>
### Open Apps and split-screen status

Your TI-89 Titanium lets you split the screen to view two Apps simultaneously. For example, view the Y= Editor and Graph screens simultaneously to see the list of functions and how they are graphed.

Select the Split Screen mode from Page 2 of the MODE screen. The TI-89 Titanium displays the selected Apps in the split-screen view as shown. Split the screen horizontally (top-bottom) or vertically (left-right).
To return to the Apps desktop, press [APPS]. The split-screen status appears at the top of the Apps desktop with the names of the open Apps and the portions of the screen in which each is displayed. The highlighted numeral indicates the split-screen portion where the next App you open will appear.

**Note:** The Apps desktop always appears in the full-screen view.
More information is available about using split screens. (For more information, see the electronic *Split Screens* chapter.)

## Checking status information

Look to the status line, located at the bottom of the screen, for information about the current state of your TI-89 Titanium.

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Current folder</td>
<td>Name of the selected folder (MAIN is the default folder.)</td>
</tr>
<tr>
<td>2 Modifier key</td>
<td>Selected modifier key (2nd, , ), if any.</td>
</tr>
<tr>
<td>3 Angle mode</td>
<td>Selected units in which angle values are displayed and interpreted (RAD, DEG, GRAD)</td>
</tr>
<tr>
<td>4 Exact/Approx mode</td>
<td>Mode in which answers are calculated and displayed (AUTO, EXACT, APPROX)</td>
</tr>
<tr>
<td>5 Graph number</td>
<td>Active of two independent graphs in split-screen mode (GR#1, GR#2)</td>
</tr>
<tr>
<td>6 Graph mode</td>
<td>Selected type of graph that can be plotted (FUNC, PAR, POL, SEQ, 3D, DE)</td>
</tr>
</tbody>
</table>
You can turn off the Apps desktop from the MODE dialog box. If you do, open Apps from the APPLICATIONS menu. To open the APPLICATIONS menu, press [APPS].

**Example:** Turn off the Apps desktop.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>![MODE button]</td>
<td>![MODE dialog box]</td>
</tr>
</tbody>
</table>

### Turning off the Apps desktop

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 Entry/Answer pairs</td>
<td>22/30—Number of entry/answer pairs (default is 30, maximum is 99) in the history area of the calculator Home screen.</td>
</tr>
<tr>
<td>8 Replace batteries</td>
<td>Displayed when batteries are low (BATT). If BATT is highlighted with a black background, change the batteries as soon as possible.</td>
</tr>
<tr>
<td>9 Busy/Pause, Locked/Archived variable</td>
<td>BUSY—Calculation or graph is in progress. PAUSE—You paused a graph or program. 🔐—Variable opened in the current editor is locked or archived and cannot be modified.</td>
</tr>
</tbody>
</table>
To turn on the Apps desktop, repeat the procedure, selecting ON in the Apps Desktop mode field. To return to the Apps desktop from the calculator Home screen, press \[\text{F3}\].

Using the clock

Use the \textbf{CLOCK} dialog box to set the time and date, select the clock display format, and turn the clock off and on.

Note: The previous open App appears (in this example, the calculator Home screen).
The clock is turned on by default. If you turn off the clock, all Clock dialog box options except Clock ON/OFF are grayed out.

Displaying the CLOCK dialog box

1. Use the cursor keys to highlight the Clock icon on the Apps desktop.
2. Press \textasciitilde. The CLOCK dialog box appears with the Time Format field highlighted.

\textbf{Note:} Because the CLOCK dialog box displays the settings current at the time you open the dialog box, you might need to update the time before exiting.

Setting the time

1. Press \textasciitilde to open the list of time formats.
2. Press \textasciitilde or \textasciitilde to highlight an option, then press \textasciitilde. The selected format appears in the Time Format field.
3. Press \textasciitilde to highlight the Hour field.
4. Type the hour, then press \textasciitilde to highlight the Minute field.
5. Type the minute(s).
6. If the time format is 24 hours, proceed to step 9.

— or —

If the time format is 12 hours, press ☰ to highlight the AM/PM field.

7. Press ☰ to open the list of AM/PM options.

8. Press ☰ or ☰ to highlight an AM/PM option, then press ENTER. The selected AM/PM option appears.

9. Set the date (for procedures, see Setting the date).

— or —

To save your settings and exit, press ENTER. The time is updated in the top right corner of the Apps desktop.

Setting the date

1. Press ☰ or ☰ as required to highlight the Date Format field.

2. Press ☰ to open the list of date formats.

3. Press ☰ or ☰ to highlight an option, then press ENTER. The selected format appears in the Date Format field.

4. Press ☰ to highlight the Year field.

5. Type the year, then press ☰ to highlight the Month field.

6. Press ☰ to open the list of months.

7. Press ☰ or ☰ to highlight an option, then press ENTER. The selected month appears in the Month field.

8. Press ☰ to highlight the Day field.
9. Type the day, then press [ENTER] [ENTER] to save your settings and exit. The date is updated in the top right corner of the Apps desktop.

**Example:** Set the time and date to 19/10/02 (October 19, 2002) at 1:30 p.m.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use cursor keys to highlight</td>
<td>Time and date [Image]</td>
</tr>
<tr>
<td>[ENTER]</td>
<td>[Image]</td>
</tr>
<tr>
<td>[1]</td>
<td>[Image]</td>
</tr>
<tr>
<td>Press</td>
<td>Result</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>3 0</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
</tr>
<tr>
<td>Press</td>
<td>Result</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td><img src="image1.png" alt="Clock Menu" /></td>
</tr>
<tr>
<td>ENTER</td>
<td><img src="image2.png" alt="Clock Menu" /></td>
</tr>
<tr>
<td>2002</td>
<td><img src="image3.png" alt="Clock Menu" /></td>
</tr>
</tbody>
</table>
Scroll down to October and press ENTER

19
### Turning off the clock

From the Apps desktop, open the CLOCK dialog box and select OFF in the Clock field.

**Example:** Turn off the clock.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTER ENTER</td>
<td>Revised time and date</td>
</tr>
</tbody>
</table>

![Clock dialog box]

Use cursor keys to highlight Clock

![Clock dialog box with Clock highlighted]

Clock on
To turn on the clock, repeat the procedure, selecting ON in the Clock field. Remember to reset the time and date.
**Using menus**

To select most TI-89 Titanium menus, press the function keys corresponding to the toolbars at the top of the calculator Home screen and most App screens. Select other menus using key commands.

**Toolbar menus**

The starting point for TI-89 Titanium math operations, the calculator Home screen displays toolbar menus that let you choose math-related options.

Toolbar menus also appear at the top of most App screens. These menus list common functions of the active App.

**Other menus**

Use key commands to select the following menus. These menus contain the same options regardless of the screen displayed or the active App.

<table>
<thead>
<tr>
<th>Press</th>
<th>To display</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd [CHAR]</td>
<td>CHAR menu. Lists characters not available on the keyboard; characters are organized by category (Greek, math, punctuation, special, and international).</td>
</tr>
<tr>
<td>2nd [MATH]</td>
<td>MATH menu. Lists math operations by category.</td>
</tr>
</tbody>
</table>
Selecting menu options

- Press the number or letter to the left of the option you want to select.
  — or —
- Press º or Û to select the option, and press ENTER.

Note: If the first menu option is selected, press º to select the last option on the menu. If the last menu option is selected, press Û to select the first option on the menu.
Example: Select `factor(` from the Algebra menu on the calculator Home screen.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Press:</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td>HOME</td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>– or –</td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td>From the Apps desktop, use the cursor keys to highlight</td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>and press <a href="image5.png">ENTER</a></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="Image" /></td>
<td>▼ indicates Algebra menu will open when you press <a href="image8.png">F2</a>.</td>
</tr>
<tr>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image11.png" alt="Image" /></td>
</tr>
<tr>
<td>– or –</td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td><a href="image13.png">ENTER</a></td>
<td><img src="image14.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Selecting submenu options

A small arrow symbol (▶) to the right of a menu option indicates that selecting the option will open a submenu.

Example: Select `ord(` from the MATH menu on the calculator Home screen.
An ellipsis (…) at the end of a menu option indicates that choosing the option will open a dialog box. Select the option and press \[\text{ENTER}\].
**Example:** Open the **SAVE COPY AS** dialog box from the Window Editor.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>APPS</strong></td>
<td><img src="image1.png" alt="Diagram" /> Use the cursor keys to highlight <strong>Window Edi...</strong> and press <strong>ENTER</strong></td>
</tr>
<tr>
<td><strong>F1</strong></td>
<td><img src="image2.png" alt="Diagram" /> Press <strong>2</strong> or <strong>ENTER</strong> to display a list of folders.</td>
</tr>
<tr>
<td><strong>2</strong></td>
<td><img src="image3.png" alt="Diagram" /> Press <strong>ENTER</strong> to display a list of folders.</td>
</tr>
<tr>
<td><strong>2</strong></td>
<td>Type the name of the variable.</td>
</tr>
<tr>
<td><strong>Enter</strong></td>
<td>Press <strong>ENTER</strong> twice to save and</td>
</tr>
</tbody>
</table>
Note: Pressing the 8 S key shortcut also opens the SAVE COPY AS dialog box in most Apps.

**Canceling a menu**

To cancel a menu without making a selection, press [ESC].

**Moving among toolbar menus**

To move among the toolbar menus without selecting a menu option:

- Press the function key (F1 through F8) of a toolbar menu.
- Press a function key, then press ‹ or › to move from one toolbar menu to the next. Press › from the last menu to move to the first menu. Press ‹ to move from the first menu to the last menu.

Note: If you press › when a menu option with a submenu is selected, the submenu will appear instead of the next toolbar menu. Press › again to move to the next menu.

More information is available about menus. (See the electronic *Operating the Calculator* chapter.)

**Custom menu**

The custom menu provides quick access to your most commonly used options. Use the default custom menu or create your own using the Program Editor. You can include any available TI-89 Titanium command or character.
The custom menu replaces the standard toolbar menu on the calculator Home screen. (For details on creating a custom menu, see the electronic Programming chapter.) More information is available about custom menus. (See the electronic Operating the Calculator chapter.)

**Example:** Turn on and turn off the custom menu from the calculator Home screen.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd [CUSTOM]</td>
<td>Default custom menu</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Default custom menu" /></td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Normal toolbar menu" /></td>
</tr>
</tbody>
</table>

**Example:** Restore the default custom menu.
**Note:** Restoring the default custom menu erases the previous custom menu. If you created the previous custom menu with a program, you can run the program again to reuse the menu.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 2nd [CUSTOM]  
(to turn off the custom menu and turn on the standard toolbar menu) |

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd [F6]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
</table>
| 3  
– or –  
▽ ▼ ENTER |

...stmOff":EndCstm;CustmOr
Opening Apps with the Apps desktop turned off

If you turn off the Apps desktop, use the APPLICATIONS menu to open Apps. To open the APPLICATIONS menu with the Apps desktop off, press \texttt{APPS}.

**Note:** If you press \texttt{APPS} with the Apps desktop turned on, the Apps desktop will appear instead of the APPLICATIONS menu.

**Example:** With the Apps desktop turned off, open the Window Editor from the APPLICATIONS menu.
To access Apps not listed on the APPLICATIONS menu, select 1:FlashApps.

**Using split screens**

The TI-89 Titanium lets you split the screen to show two Apps at the same time. For example, display both the Y= Editor and Graph screens to compare the list of functions and how they are graphed.

**Setting split-screen mode**

You can split the screen either top to bottom or left to right from the MODE dialog box. The split-screen setting stays in effect until you change it.

1. Press **MODE** to display the MODE dialog box.
2. Press **F2** to display the Split Screen mode setting.
3. Press **◊** to open the Split Screen mode menu.
4. Press **◊** as required to highlight either TOP-BOTTOM or LEFT-RIGHT.
5. Press **ENTER**. The Split Screen mode setting displays the option you selected.
6. Press **ENTER** again to save this change and display the split screen.
**Example:** Set split-screen mode to TOP-BOTTOM.

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE</td>
<td><img src="image1" alt="MODE Setting" /></td>
</tr>
<tr>
<td>F2</td>
<td><img src="image2" alt="F2 Setting" /></td>
</tr>
<tr>
<td>1/2</td>
<td><img src="image3" alt="Split Screen" /></td>
</tr>
</tbody>
</table>
Setting the initial Apps for split screen

After you select either TOP-BOTTOM or LEFT-RIGHT split-screen mode, additional mode settings become available.
To set the initial App for each split-screen portion:

1. Select the Split 1 App mode setting and press B to display a menu of available Apps. (See “Setting split-screen mode” on page 59.)
2. Press C or D to highlight the App and press ENTER.
3. Repeat steps 1 and 2 for the Split 2 App mode setting.

**Example:** Display the Y= Editor in the top screen and the Graph App in the bottom screen.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split 2 App</td>
<td>Lets you specify the App displayed in the bottom or right portion of the split screen. Works together with Split 1 App, which lets you specify the App displayed in the top or left portion of the split screen.</td>
</tr>
<tr>
<td>Number of Graphs</td>
<td>Lets you set up and display two independent graphs.</td>
</tr>
</tbody>
</table>
### Getting Started

<table>
<thead>
<tr>
<th>Press</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td><img src="image1.png" alt="Press 2 diagram" /></td>
</tr>
<tr>
<td>4</td>
<td><img src="image2.png" alt="Press 4 diagram" /></td>
</tr>
<tr>
<td>ENTER</td>
<td><img src="image3.png" alt="Enter diagram" /></td>
</tr>
</tbody>
</table>
If you set Split 1 App and Split 2 App to the same nongraphing App or to the same graphing App with Number of Graphs set to 1, the TI-89 Titanium exits split-screen mode and displays the App in full-screen mode.

**Selecting the active App**

In split-screen mode, only one App can be active at a time.

- To switch between active Apps, press \texttt{2nd} \texttt{[++]}. 
- To open a third App, press \texttt{APPS} and select the App. This App replaces the active split-screen App.

**Exiting split-screen mode**

Exit split-screen mode in any of the following ways:

- Press \texttt{2nd} \texttt{[QUIT]} to close the active App and display the full-screen view of the other open App.
- If the Apps desktop is turned off, pressing \texttt{2nd} \texttt{[QUIT]} replaces the active split-screen App with the calculator Home screen. Pressing \texttt{2nd} \texttt{[QUIT]} again turns off the split-screen mode and displays the calculator Home screen in full-screen mode.
- Select Split Screen on Page 2 of the MODE dialog box, set split-screen mode to FULL, and press \texttt{ENTER}.
- Press \texttt{2nd} \texttt{[QUIT]} twice to display the Apps desktop

More information is available about using split screens. (See the electronic *Split Screens* chapter.)
Managing Apps and operating system (OS) versions

Using the TI-89 Titanium connectivity features, you can download Apps from:

- The TI Educational & Productivity Solutions (E&PS) Web site at: education.ti.com/latest
- The CD-ROM included with your TI-89 Titanium.
- A compatible graphing calculator.

Adding Apps to your TI-89 Titanium is like loading software on a computer. All you need is TI Connect™ software and the USB computer cable that came with your TI-89 Titanium.

For system requirements and instructions to link to compatible calculators and download TI Connect software, Apps, and OS versions, see the TI E&PS Web site.

Before downloading Apps to your TI-89 Titanium, please read the license agreement on the CD-ROM or TI Web site.

Finding the OS version and identification (ID) numbers

If you purchase software from the TI E&PS Web site or call the customer support number, you will be asked to provide information about your TI-89 Titanium. You will find this information on the ABOUT screen.
To display the ABOUT screen, press [F1] 3:About from the Apps desktop. The ABOUT screen displays the following information about your TI-89 Titanium:

1. OS version

2. Hardware version

3. Unit ID (required to obtain certificates for installing purchased Apps). Similar to a serial number. Write this number down and keep it in a safe place in case the calculator is ever lost or stolen.

4. Apps certificate revision number (Cert. Rev.)

5. Product identifier (Product ID). Similar to a model number.

Note that your screen will be different than the one shown above.
Deleting an Application

Deleting an application removes it from the TI-89 Titanium and increases space for other applications. Before deleting an application, consider storing it on a computer for reinstallation later.

1. Quit the application.
2. Press [2nd] [VAR-LINK] to display the VAR-LINK (All) screen.
3. Press [2nd] [F7] to display the list of installed applications.
4. Select the application you want to delete by pressing [F4]. (Press [F4] again to deselect.)
6. Press [ENTER] to delete the application.

Note: Only Flash Apps can be deleted.

Connecting your TI-89 Titanium to other devices

The TI-89 Titanium includes both a mini-USB port and a standard I/O port. Ports are used to link two compatible graphing calculators or connect to a computer or peripheral device.

In addition, the teacher model of the TI-89 Titanium includes an accessory port. This port is used to output visual data so that a classroom can view the calculator’s display on a video device or overhead screen.

To connect your calculator to a computer – Connect your TI-89 Titanium using the USB port and the included USB computer cable.
To connect your calculator to another calculator – Use the USB unit-to-unit cable or an I/O unit-to-unit cable to connect the TI-89 Titanium to a compatible graphing calculator or peripheral device, such as a TI-89 or TI-92 Plus graphing calculator or the CBL 2™ and CBR™ systems.

To show your calculator’s display to the classroom – Use the accessory port to connect the TI-Presenter™ video adapter to the teacher model of the TI-89 Titanium. The TI-Presenter video adapter provides a video interface between the calculator and video display or recording devices. Or use the accessory port to connect the TI ViewScreen™ overhead panel to your calculator. The TI ViewScreen overhead panel enlarges and projects the display so an entire class can view it. For more information about the TI-Presenter video adapter and TI ViewScreen panel, see the TI E&PS Web site at education.ti.com.

![TI-89 Titanium ports](image-url)
Batteries

The TI-89 Titanium uses four AAA alkaline batteries and a backup silver oxide battery (SR44SW or 303). The backup battery is already installed, and the AAA batteries are provided with your product.

Important OS download information

New batteries should be installed before beginning an OS download.

When in OS download mode, the APD™ feature does not function. If you leave your calculator in download mode for an extended time before you actually start the download, your batteries may become depleted. You will then need to replace the depleted batteries with new batteries before downloading.
You can also transfer the OS to another TI-89 Titanium using a USB unit-to-unit cable. If you accidentally interrupt the transfer before it is complete, you will need to reinstall the OS via a computer. Again, remember to install new batteries before downloading.

Please contact Texas Instruments as described in Service & Support Information, if you experience a problem.

**Battery Precautions**

Take these precautions when replacing batteries:

- Do not leave batteries within the reach of children.
- Do not mix new and used batteries. Do not mix brands (or types within brands) of batteries.
- Do not mix rechargeable and non-rechargeable batteries.
- Install batteries according to polarity (+ and –) diagrams.
- Do not place non-rechargeable batteries in a battery recharger.
- Properly dispose of used batteries immediately.
- Do not incinerate or dismantle batteries.

**Installing the AAA Batteries**

1. Remove the battery cover from the back of the calculator.
2. Unwrap the four AAA batteries provided with your product and insert them in the battery compartment. Arrange the batteries according to the polarity (+ and –) diagram in the battery compartment.
3. Replace the battery cover on the calculator. The cover should snap into place.

**Replacing the AAA (alkaline) batteries**

As the batteries lose power, the display begins to dim, especially during calculations. If you find yourself increasing the contrast frequently, replace the AAA alkaline batteries.

The status line also gives battery information.

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BATT</td>
<td>Batteries are low.</td>
</tr>
<tr>
<td>BATT</td>
<td>Replace batteries as soon as possible.</td>
</tr>
</tbody>
</table>

Before replacing the batteries, turn off the TI-89 Titanium by pressing 2nd [OFF] to avoid losing information stored in memory. Do not remove both the back-up battery and the AAA alkaline batteries at the same time.
Replacing the backup (silver oxide) battery

1. To replace the silver oxide backup battery, remove the battery cover and unscrew the tiny screw holding the BACK UP BATTERY cover in place.

2. Remove the old battery and install a new SR44SW or 303 battery, positive (+) side up. Replace the cover and the screw.
Performing Computations

This section provides several examples for you to perform from the Calculator Home screen that demonstrate some of the computational features of the TI-89 Titanium. The history area in each screen was cleared by pressing F1 and selecting 8:Clear Home, before performing each example, to illustrate only the results of the example’s keystrokes.

Showing Computations

<table>
<thead>
<tr>
<th>Steps and keystrokes</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute $\sin(\pi/4)$ and display the result in symbolic and numeric format. To clear the history area of previous calculations, press F1 and select 8:Clear Home.</td>
<td>![Display Image]</td>
</tr>
</tbody>
</table>

Press 2nd [SIN] 2nd [π] ÷ 4 ENTER [≈]
Finding the Factorial of Numbers

Steps and keystrokes

Compute the factorial of several numbers to see how the TI-89 Titanium handles very large integers. To get the factorial operator (!), press 2nd [MATH], select 7:Probability, and then select 1:!. 

Press 5 2nd [MATH] 7 1 ENTER 20 2nd [MATH] 7 1 ENTER 30 2nd [MATH] 7 1 ENTER

Expanding Complex Numbers

Steps and keystrokes

Compute \((3+5i)^3\) to see how the TI-89 Titanium handles computations involving complex numbers.

Press ( 3 + 5 2nd [i] ) ) ^ 3 ENTER
Finding Prime Factors

Steps and keystrokes

Compute the factors of the rational number 2634492. You can enter “factor” on the entry line by typing `FACTOR` on the keyboard, or by pressing `F2` and selecting `2:factor`.

Press `F2 2 2634492 ) ENTER`.

(Optional) Enter other numbers on your own.

Finding Roots

Steps and keystrokes

Find the root of the expression `(x,y)`. You can enter “root” on the entry line by typing `ROOT` on the keyboard, or by pressing `9`.

This example illustrates using the root function and how the expression is displayed in “pretty print” in the history area.

Press `9 X Y ) ENTER`.
Expanding Expressions

Steps and keystrokes

Expand the expression \((x-5)^3\). You can enter “expand” on the entry line by typing \textbf{EXPAND} on the keyboard, or by pressing \textbf{F2} and selecting \textbf{3:expand(}.

Press \textbf{F2} 3 \(\langle \text{x} \text{ } | \text{ } - \text{ } 5 \rangle \hat{\text{x}} 3 \rangle \) \text{ ENTER}

\textit{(Optional)} Enter other expressions on your own.

Reducing Expressions

Steps and keystrokes

Reduce the expression \((x^2-2x-5)/(x-1)\) to its simplest form. You can enter “propFrac” on the entry line by typing \textbf{PROPFRAC} on the keyboard, or by pressing \textbf{F2} and selecting \textbf{7:propFrac(}.

Press \textbf{F2} 7 \(\langle \text{x} \hat{\text{x}} 2 \rangle \text{ } - \text{ } 2 \text{ } \langle \text{x} \rangle \text{ } - \text{ } 5 \rangle \div \langle \text{x} \rangle 1 \rangle \text{ ENTER}
Factoring Polynomials

Steps and keystrokes

Factor the polynomial \((x^2 - 5)\) with respect to \(x\).
You can enter “factor” on the entry line by
typing **FACTOR** on the keyboard or by
pressing **F2** and selecting **2:factor**.

Press **F2** 2 X \(^2\) 2 - 5 \(\text{ }\) X \(\text{ }\) ENTER

Solving Equations

Steps and keystrokes

Solve the equation \(x^2 - 2x - 6 = 2\) with respect to \(x\).
You can enter “**solve**” on the entry line by
selecting “**solve**” from the Catalog menu, by
typing **SOLVE** on the keyboard, or by
pressing **F2** and selecting **1:solve**.

The status line area shows the required syntax for the marked item in the **Catalog** menu.

Press **F2** 1 X \(^2\) 2 - 2 X \(-\) 6 \(\text{ }\) 2 \(\text{ }\) X \(\text{ }\) ENTER
Solving Equations with a Domain Constraint

Steps and keystrokes

Solve the equation \(x^2 - 2x - 6 = 2\) with respect to \(x\) where \(x\) is greater than zero. The "with" (I) operator provides domain constraint.

Press \(\text{F2} \ 1 \ X \ 2 \ - \ 2 \ X \ - \ 6 \ = \ 2 \ , \ X \ ) \ \text{I} \ X \ \text{2nd} \ [ \ > \ ] \ 0 \ \text{ENTER} \)

Display

\[
\text{solve}(x^2 - 2x - 6 = 2, x) | x > 0
\]
\[x = 4\]

Solving Inequalities

Steps and keystrokes

Solve the inequality \((x^2 > 1, x)\) with respect to \(x\).

Press \(\text{F2} \ 1 \ X \ ^2 \ 2 \ \text{2nd} \ [ \ > \ ] \ 1 \ \text{I} \ \text{ENTER} \)

Display

\[
\text{solve}(x^2 > 1, x)
\]
\[x < -1 \text{ or } x > 1\]
**Finding the Derivative of Functions**

**Steps and keystrokes**

Find the derivative of \((x-y)^3/(x+y)^2\) with respect to \(x\).

This example illustrates using the calculus differentiation function and how the function is displayed in “pretty print” in the history area.

Press `2nd [d] ( X - Y ) ^ 3 ÷ ( X + Y ) ^ 2` ENTER

**Display**

\[
\frac{\partial}{\partial x} \left( \frac{(x-y)^3}{(x+y)^2} \right) = \frac{(x-y)^2 \cdot (x+5 \cdot y)}{(x+y)^3}
\]

**Finding Implicit Derivatives**

**Steps and keystrokes**

Compute implicit derivatives for equations in two variables in which one variable is defined implicitly in terms of another.

This example illustrates using the calculus implicit derivative function.

Press `F3 D X ^ 2 + Y ^ 2 = 100 , X , Y` ENTER
### Finding the Integral of Functions

**Steps and keystrokes**

Find the integral of $x \cdot \sin(x)$ with respect to $x$.

This example illustrates using the calculus integration function.

Press $2\text{nd} [\int] X \times 2\text{nd} [\sin] X ) , X ) ) \text{ ENTER}$

<table>
<thead>
<tr>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\int (x \cdot \sin(x)) , dx$</td>
</tr>
<tr>
<td>$\sin(x) - x \cdot \cos(x)$</td>
</tr>
<tr>
<td>$F(x \cdot \sin(x), x)$</td>
</tr>
</tbody>
</table>

### Solving Problems Involving Vectors

**Steps and keystrokes**

1. **Input a row or column of vectors.**

   Press $2\text{nd} [,] \ (-) 6 , 0 , 0 2\text{nd} [,] \text{ STO}\alpha d \text{ ENTER} 2\text{nd} [,] 4 , 0 , 2 2\text{nd} [,] \text{ STO}\alpha a \text{ ENTER} 2\text{nd} [,] (-) 1 , 2$
   
   $, 1 2\text{nd} [,] \text{ STO}\alpha b \text{ ENTER} 2\text{nd} [,] \text{ STO}\alpha c \text{ ENTER}$

<table>
<thead>
<tr>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[-6 \ 0 \ 0] \rightarrow d$</td>
</tr>
<tr>
<td>$[4 \ 0 \ 2] \rightarrow a$</td>
</tr>
<tr>
<td>$[-1 \ 2 \ 1] \rightarrow b$</td>
</tr>
<tr>
<td>$[7 \ 6 \ 5] \rightarrow c$</td>
</tr>
<tr>
<td>$[[7,6,5]] \rightarrow d$</td>
</tr>
</tbody>
</table>

2. **Solve (x*a+y*b+z*c=d \{x,y,z\})**

   Press $F2 1 X \times \alpha a + y \times \alpha b + z \times \alpha c = \alpha d , 2\text{nd} [,] X , Y , Z 2\text{nd} [,] ) ) \text{ ENTER}$

<table>
<thead>
<tr>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[-6 \ 0 \ 0] \rightarrow d$</td>
</tr>
<tr>
<td>$[4 \ 0 \ 2] \rightarrow a$</td>
</tr>
<tr>
<td>$[-1 \ 2 \ 1] \rightarrow b$</td>
</tr>
<tr>
<td>$[7 \ 6 \ 5] \rightarrow c$</td>
</tr>
<tr>
<td>$solve(x<em>a+y</em>b+z*c=d, (x=1 \ and \ y=3 \ and \ z=-1)$</td>
</tr>
<tr>
<td>$e(x<em>a+y</em>b+z*c=d, (x,y,z))$</td>
</tr>
<tr>
<td>$\text{MAIN} \quad \text{GRD AUTO} \quad \text{FUNK} \quad 4/20$</td>
</tr>
</tbody>
</table>
## Log to Any Base

### Steps and keystrokes  Display

Find log (x,b). You can enter “log” on the entry line by typing \textbf{LOG} on the keyboard, or by pressing $\blacktriangleleft$ 7.

Press $\blacktriangleleft$ 7 X , \texttt{alpha} b $\blacktriangleright$ ENTER

### Converting Angle Measures

#### Steps and keystrokes  Display

1. Display the \textbf{MODE} dialog box. For \textbf{Angle} mode select \textbf{DEGREE}. Convert 345 degrees to Gradian angle measure.

You can enter “\textbf{Grad}” on the entry line by selecting “\textbf{Grad}” from the Catalog menu, or from the Math menu by pressing $\blacktriangledown$ [MATH] and selecting 2:angle, A:\textbf{Grad}.

Press MODE $\blacktriangledown$ $\blacktriangledown$ $\blacktriangledown$ $\blacktriangledown$ 2 ENTER 345 $\blacktriangledown$ [°] 2nd [MATH] 2 \texttt{alpha} A ENTER
Symbolic Manipulation

Solve the system of equations $2x - 3y = 4$ and $-x + 7y = -12$. Solve the first equation so that $x$ is expressed in terms of $y$. Substitute the expression for $x$ into the second
equation, and solve for the value of $y$. Then substitute the $y$ value back into the first equation to solve for the value of $x$.

### Steps and keystrokes

<table>
<thead>
<tr>
<th>Steps</th>
<th>Display</th>
</tr>
</thead>
</table>
| **1.** Display the Home screen and clear the entry line. Solve the equation $2x - 3y = 4$ for $x$.  

**F2** 1 selects `solve( from the Algebra menu. You can also type `solve( directly from the keyboard or select it from the Catalog.  

Press **HOME** **CLEAR** **CLEAR** **F2** 1 2 X − 3 Y ≡ 4 , X ) **ENTER**  

**2.** Begin to solve the equation $-x + 7y = -12$ for $y$, but do not press **ENTER** yet.  

Press **F2** 1 (−) X + 7 Y ≡ (−) 12 , Y ) |
| **3.** Use the “*with*” operator to substitute the expression for $x$ that was calculated from the first equation. This gives the value of $y$.  

The “*with*” operator is displayed as / on the screen.  

Use the auto-paste feature to highlight the last answer in the history area and paste it to the entry line.  

Press **1** **(−)** **ENTER** **ENTER** |
Steps and keystrokes | Display
--- | ---
4. Highlight the equation for x in the history area. | ![Display 1](image)
   - Press ▲ ▲ ▲

5. Auto-paste the highlighted expression to the entry line. Then substitute the value of y that was calculated from the second equation. | ![Display 2](image)
   - Press ENTER 1 ▲ ENTER ENTER
   - The solution is: x = -8/11 and y = -20/11

This example is a demonstration of symbolic manipulation. A one-step function is available for solving systems of equations.

**Constants and Measurement Units**

Using the equation f = m*a, calculate the force when m = 5 kilograms and a = 20 meters/second\(^2\). What is the force when a = 9.8 meters/second\(^2\). (This is the
acceleration due to gravity, which is a constant named \( g \). Convert the result from newtons to kilograms of force.

### Steps and keystrokes

<table>
<thead>
<tr>
<th>Steps</th>
<th>Display</th>
</tr>
</thead>
</table>
| 1. Display the **MODE** dialog box, Page 3. For **Unit System** mode, select **SI** for the metric system of measurements. Results are displayed according to these default units. <br>Press **MODE** **F3** **1** **ENTER** | ![Mode Menu]
| 2. Create an acceleration unit for meters/second\(^2\) named \( \text{ms}^2 \). <br>The **UNITS** dialog box lets you select units from an alphabetical list of categories. You can use **2nd** **and** **2nd** to scroll one page at a time through the categories. <br>If you use the **UNITS** dialog box to select a unit, the \( _{} \) is entered automatically. Now, instead of re-entering \( _m/_s^2 \) each time you need it, you can use \( _\text{ms}^2 \). Also, you can now use the **UNITS** dialog box to select \( _\text{ms}^2 \) from the Acceleration category. <br>Press **2nd** **[UNITS]** **\( _{} \) M** **ENTER** **\( _{} \) 2** **ENTER** | ![Units Menu]
| ![Units Menu](image)
| ![Units Menu](image)
| ![Units Menu](image) | ![Units Menu](image)

Press \( _{} \) **2nd** **[UNITS]** **\( _{} \) M** **ENTER** **\( _{} \) 2** **ENTER** **\( _{} \) [a-lock] MS** **alpha 2** **ENTER** | ![Units Menu](image)
Previews 86

Basic Function Graphing I

The example in this section demonstrates some of the graphing capabilities of the TI-89 Titanium keystrokes. It illustrates how to graph a function using the Y= Editor. You will
learn how to enter a function, produce a graph of the function, trace a curve, find a minimum point, and transfer the minimum coordinates to the Home screen.

Explore the graphing capabilities of the TI-89 Titanium by graphing the function \( y = (|x^2 - 3| - 10)/2 \).

### Steps and keystrokes

<table>
<thead>
<tr>
<th>Steps and keystrokes</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Display the <strong>Y= Editor</strong>.</td>
<td><img src="image1" alt="Entry Line" /></td>
</tr>
<tr>
<td>Press ( \downarrow ) [Y=]</td>
<td><img src="image2" alt="Main Screen" /></td>
</tr>
</tbody>
</table>

2. Enter the function \( (\text{abs}(x^2 - 3) - 10)/2 \).

   The screen shot shows the “pretty print” display at \( y_1 = \).

   Press \( ( \) \text{CATALOG} \) A \( \) \text{ENTER} \( X^2 \) \( - \) 3 \( ) \(-\) 1 \( 0 \) \( ) \div 2 \) \text{ENTER} |

3. Display the graph of the function.

   Select **6:ZoomStd** by pressing 6 or by moving the cursor to **6:ZoomStd** and pressing \( \text{ENTER} \).

   Press \( \text{F2} \) 6
### Steps and keystrokes

<table>
<thead>
<tr>
<th>Steps</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. <strong>Turn on Trace.</strong>  &lt;br&gt; The tracing cursor, and the x and y coordinates are displayed.  &lt;br&gt; Press F3</td>
<td><img src="image1" alt="Tracing cursor" /></td>
</tr>
<tr>
<td>5. <strong>Open the MATH menu and select 3:Minimum.</strong>  &lt;br&gt; Press F5 → → ENTER</td>
<td><img src="image2" alt="Math menu" /></td>
</tr>
<tr>
<td>6. <strong>Set the lower bound.</strong>  &lt;br&gt; Press → (right cursor) to move the tracing cursor until the lower bound for x is just to the left of the minimum node before pressing ENTER the second time.  &lt;br&gt; Press → ... → ENTER</td>
<td><img src="image3" alt="Lower Bound" /></td>
</tr>
<tr>
<td>7. <strong>Set the upper bound.</strong>  &lt;br&gt; Press → (right cursor) to move the tracing cursor until the upper bound for x is just to the right of the minimum node.  &lt;br&gt; Press → ... →</td>
<td><img src="image4" alt="Upper Bound" /></td>
</tr>
</tbody>
</table>
Basic Function Graphing II

Graph a circle of radius 5, centered on the origin of the coordinate system. View the circle using the standard viewing window (ZoomStd). Then use ZoomSqr to adjust the viewing window.

Steps and keystrokes

1. Display the MODE dialog box. For Graph mode, select FUNCTION.
   
   Press MODE 1 ENTER
<table>
<thead>
<tr>
<th>Steps and keystrokes</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Display the Home screen. Then store the radius, 5, in variable r.</td>
<td>5⇒r</td>
</tr>
<tr>
<td>Press <strong>HOME</strong> 5 <strong>STO►</strong> [alpha] R <strong>ENTER</strong></td>
<td></td>
</tr>
<tr>
<td>3. Display and clear the <strong>Y= Editor</strong>. Then define ( y_1(x) = \sqrt{r^2 - x^2} ), the top half of a circle.</td>
<td></td>
</tr>
<tr>
<td>In function graphing, you must define separate functions for the top and bottom halves of a circle.</td>
<td></td>
</tr>
<tr>
<td>Press • [Y=] F1 8 <strong>ENTER</strong> <strong>ENTER</strong> 2nd [√] [alpha] R (^2) (-) X (^2) <strong>ENTER</strong></td>
<td></td>
</tr>
<tr>
<td>4. Define ( y_2(x) = -\sqrt{r^2 - x^2} ), the function for the bottom half of the circle.</td>
<td></td>
</tr>
<tr>
<td>The bottom half is the negative of the top half, so you can define ( y_2(x) = -y_1(x) ).</td>
<td></td>
</tr>
<tr>
<td>Use the full function name ( y_1(x) ), not simply ( y_1 ).</td>
<td></td>
</tr>
<tr>
<td>Press <strong>ENTER</strong> (-) Y 1 (X) <strong>ENTER</strong></td>
<td></td>
</tr>
</tbody>
</table>
5. Select the **ZoomStd** viewing window, which automatically graphs the functions.

In the standard viewing window, both the x and y axes range from -10 to 10. However, this range is spread over a longer distance along the x axis than the y axis. Therefore, the circle appears as an ellipse.

Press **F2** 6

6. Select **ZoomSqr**.

**ZoomSqr** increases the range along the x axis so that circles and squares are shown in correct proportion.

Press **F2** 5

**Note:** There is a gap between the top and bottom halves of the circle because each half is a separate function. The mathematical endpoints of each half are (-5,0) and (5,0). Depending on the viewing window, however, the *plotted* endpoints for each half may be slightly different from their *mathematical* endpoints.
Use the “Detect Discontinuities” graph format to eliminate faux asymptotes and connections in a jump discontinuity.

### Steps and keystrokes

<table>
<thead>
<tr>
<th>Steps</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Display the <strong>MODE</strong> dialog box. For <strong>Graph</strong> mode, select <strong>FUNCTION</strong>. For <strong>Angle</strong> mode, select <strong>RADIANS</strong>.</td>
<td></td>
</tr>
<tr>
<td>Press <strong>MODE</strong> ➤ 1 ➤ 1 ➤ 1 ➤ 1 ENTER</td>
<td></td>
</tr>
<tr>
<td>2. Open the <strong>Y=</strong> Editor and enter $y_1(x) = 1/(x-1)$.</td>
<td></td>
</tr>
<tr>
<td>Press <strong>[Y=]</strong> ➤ 1 ➤ ( X ➤ 1 ➤ ) ➤ ENTER</td>
<td></td>
</tr>
<tr>
<td>3. Display the Graph Formats dialog box and set “Detect Discontinuities” to OFF</td>
<td></td>
</tr>
<tr>
<td><strong>Note</strong>: The second item on the Graph Format dialog is not greyed out, which means it can be set to sequential “Seq” or simultaneous “Simul”.</td>
<td></td>
</tr>
<tr>
<td>Press <strong>[</strong> ➤ 1 ➤ 1 ➤ 1 ➤ 1 ➤ 1 ➤ ENTER</td>
<td></td>
</tr>
</tbody>
</table>
4. Execute the **Graph** command, which automatically displays the Graph screen. Observe the “faux” asymptotes contained in the graph.

Press **[GRAPH]**

<table>
<thead>
<tr>
<th>Steps and keystrokes</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>4.</strong> Execute the <strong>Graph</strong> command, which automatically displays the Graph screen. Observe the “faux” asymptotes contained in the graph.</td>
<td><img src="image" alt="Graph Screen" /></td>
</tr>
<tr>
<td>Press <strong>[GRAPH]</strong></td>
<td><img src="image" alt="Graph Formats Dialog Box" /></td>
</tr>
</tbody>
</table>

5. Display the Graph Formats dialog box and set “Detect Discontinuities” to ON.

**Note:** The second item on the Graph Format dialog is greyed out, which means the graph order is set to sequential “Seq”.

Press **[GRAPH]**

<table>
<thead>
<tr>
<th>Steps and keystrokes</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>5.</strong> Display the Graph Formats dialog box and set “Detect Discontinuities” to ON.</td>
<td><img src="image" alt="Graph Formats Dialog Box" /></td>
</tr>
<tr>
<td><strong>Note:</strong> The second item on the Graph Format dialog is greyed out, which means the graph order is set to sequential “Seq”. Press <strong>[GRAPH]</strong></td>
<td><img src="image" alt="Graph Formats Dialog Box" /></td>
</tr>
</tbody>
</table>

6. Execute the **Graph** command, which automatically displays the Graph screen. No”faux” asymptotes are present on the graph.

**Note:** Graphing speed may slow considerably when “Detect Discontinuities” is set to ON.

Press **[GRAPH]**

<table>
<thead>
<tr>
<th>Steps and keystrokes</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>6.</strong> Execute the <strong>Graph</strong> command, which automatically displays the Graph screen. No”faux” asymptotes are present on the graph.</td>
<td><img src="image" alt="Graph Screen" /></td>
</tr>
<tr>
<td><strong>Note:</strong> Graphing speed may slow considerably when “Detect Discontinuities” is set to ON. Press <strong>[GRAPH]</strong></td>
<td><img src="image" alt="Graph Screen" /></td>
</tr>
</tbody>
</table>
Graph the parametric equations describing the path of a ball kicked at an angle ($\theta$) of $60^\circ$ with an initial velocity ($v_0$) of 15 meters/sec. The gravity constant $g = 9.8$ meters/sec$^2$. Ignoring air resistance and other drag forces, what is the maximum height of the ball and when does it hit the ground?

**Steps and keystrokes**

1. **Display the MODE dialog box.** For **Graph** mode, select **PARAMETRIC**.

   Press **MODE** $\Rightarrow$ 2 **ENTER**

2. **Display and clear the Y= Editor.** Then define the horizontal component $x_{t1}(t) = v_0 t \cos \theta$.

   Enter values for $v_0$ and $\theta$.

   Press $\uparrow$ **[Y=]** **F1** 8 **ENTER** **ENTER** 15 **T** $\times$

   $\text{2nd} [\text{COS}]$ 60 $\text{2nd} [^\circ]$ $\theta$ **ENTER**

   Type $T \times \text{2nd} [\text{COS}]$, not $T \text{2nd} [\text{COS}]$.

   Enter a $^\circ$ symbol by typing either $\text{2nd} [^\circ]$ or $\text{2nd} [\text{MATH}]$ 2 1. This ensures a number is interpreted as degrees, regardless of the angle mode.
3. Define the vertical component
\[ y(t) = v_0 t \sin \theta - \frac{g}{2} t^2. \]
Enter values for \( v_0, \theta, \) and \( g. \)
Press \( \text{[ENTER]} \) \( 15 \times \text{[2nd]} \) \( \text{[SIN]} \) \( 60 \text{[2nd]} \) \( \text{[°]} \) \( \text{[)} \) \(-\) \( 9.8 \div 2 \) \( \text{T} \) \( ^2 \) \( \text{[ENTER]} \)

4. Display the **Window Editor.** Enter Window variables appropriate for this example.

You can press either \( \text{[]} \) or \( \text{[ENTER]} \) to enter a value and move to the next variable.

Press \( \text{[]} \) \( \text{[WINDOW]} \) \( 0 \) \( 3 \) \( .02 \) \( (-) \) \( 2 \) \( 25 \) \( 5 \) \( (-) \) \( 2 \) \( 10 \) \( (-) \) \( 5 \)

5. Graph the parametric equations to model the path of the ball.
Press \( \text{[]} \) \( \text{[GRAPH]} \)

6. Select **Trace.** Then move the cursor along the path to find the:
   - \( y \) value at maximum height.
   - \( t \) value where the ball hits the ground.

Press \( \text{[F3]} \) \( \text{[]} \) or \( \text{[]} \) as necessary
Polar Graphing

The graph of the polar equation \( r_1(\theta) = A \sin B\theta \) forms the shape of a rose. Graph the rose for \( A=8 \) and \( B=2.5 \). Then explore the appearance of the rose for other values of \( A \) and \( B \).

### Steps and keystrokes

<table>
<thead>
<tr>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MODE</strong> dialog box. For <strong>Graph</strong> mode, select <strong>POLAR</strong>. For <strong>Angle</strong> mode, select <strong>RADIANS</strong>.</td>
</tr>
<tr>
<td>Press <strong>MODE</strong> ( \downarrow ) 3 ( \downarrow ) ( \downarrow ) ( \downarrow ) ( \downarrow ) 1 <strong>ENTER</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Y= Editor</strong>. Then define the polar equation ( r_1(\theta) = A \sin B\theta ).</td>
</tr>
<tr>
<td>Enter 8 and 2.5 for ( A ) and ( B ), respectively.</td>
</tr>
<tr>
<td>Press ( \downarrow ) [<strong>Y=</strong>] F1 8 <strong>ENTER</strong> ENTER 8 2nd [<strong>SIN</strong>] 2.5 ( \downarrow ) [( \theta ) ] ) <strong>ENTER</strong></td>
</tr>
</tbody>
</table>
3. Select the **ZoomStd** viewing window, which graphs the equation.

- The graph shows only five rose petals.
  - In the standard viewing window, the Window variable $\theta_{\text{max}} = 2\pi$.
    The remaining petals have $\theta$ values greater than $2\pi$.

- The rose does not appear symmetrical.
  - Both the $x$ and $y$ axes range from -10 to 10. However, this range is spread over a longer distance along the $x$ axis than the $y$ axis.

Press **F2** 6

4. Display the **Window Editor**, and change $\theta_{\text{max}}$ to $4\pi$.

$4\pi$ will be evaluated to a number when you leave the **Window Editor**.

Press $\diamondsuit$ [WINDOW] $\circlearrowleft$ 4 [2nd] [$\pi$]
A small forest contains 4000 trees. Each year, 20% of the trees will be harvested (with 80% remaining) and 1000 new trees will be planted. Using a sequence, calculate the number of trees in the forest at the end of each year. Does it stabilize at a certain number?

<table>
<thead>
<tr>
<th>Initially</th>
<th>After 1 Year</th>
<th>After 2 Years</th>
<th>After 3 Years</th>
<th>. . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>.8 x 4000 + 1000</td>
<td>.8 x (.8 x 4000 + 1000) + 1000</td>
<td>.8 x (.8 x (.8 x 4000 + 1000) + 1000) + 1000</td>
<td>. . .</td>
</tr>
</tbody>
</table>
Steps and keystrokes

1. Display the **MODE** dialog box. For **Graph** mode, select **SEQUENCE**.
   
   Press `[MODE] [4] ENTER`

2. Display and clear the **Y=** **Editor**. Then define the sequence as
   
   \[ u_1(n) = \text{iPart}(0.8u_1(n-1)+1000) \]

   Use **iPart** to take the integer part of the result. No fractional trees are harvested.

   To access **iPart**(, you can use `[2nd] [MATH]`, simply type it, or select it from the **CATALOG**.
   
   Press `[Y=] F1 8 ENTER ENTER 2nd [MATH] 1 4 . 8 \text{alpha} U1 ( ( \text{alpha} \text{N} - 1 ) ) + 1000 \text{alpha} ] ENTER`

3. Define \( u_1 \) as the initial value that will be used as the first term.
   
   Press `ENTER 4000 ENTER`

4. Display the **Window Editor**. Set the \( n \) and plot **Window variables**.

   \( n_{\text{min}}=0 \) and \( n_{\text{max}}=50 \) evaluate the size of the forest over 50 years.

   Press `[WINDOW] 0 \text{down} 50 \text{down} 1 \text{up} 1 \text{up}`
5. Set the x and y Window variables to appropriate values for this example.

   Press 0 ⇐ 50 ⇐ 10 ⇐ 0 ⇐ 6000 ⇐ 1000

6. Display the Graph screen.

   Press [GRAPH]

7. Select Trace. Move the cursor to trace year by year. How many years (nc) does it take the number of trees (yc) to stabilize?

   Trace begins at nc=0.

   nc is the number of years.

   xc = nc since n is plotted on the x axis.

   yc = u1(n), the number of trees at year n.

   Press F3 ⇐ and ↑ as necessary
3D Graphing

Graph the 3D equation $z(x,y) = (x^3y - y^3x) / 390$. Animate the graph by using the cursor to interactively change the eye Window variable values that control your viewing angle. Then view the graph in different graph format styles.

### Steps and keystrokes

1. Display the **MODE** dialog box. For **Graph** mode, select **3D**.
   
   Press **MODE** → 5 **ENTER**

2. Display and clear the **Y= Editor**. Then define the 3D equation
   
   $z_1(x,y) = (x^3y - y^3x) / 390$.
   
   Notice that implied multiplication is used in the keystrokes.
   
   Press **♦ [Y=] F1 8 ENTER ENTER ( X ^ 3 Y - Y ^ 3 X ) / 390 ENTER**

3. Change the graph format to display and label the axes. Also set **Style = WIRE FRAME**.

   You can animate any graph format style, but **WIRE FRAME** is fastest.

   Press **♦ ♠ ♦ • 2 ♦ • 2 ♦ • 1 ENTER**
### Steps and keystrokes

<table>
<thead>
<tr>
<th>Steps</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Select the <strong>ZoomStd</strong> viewing cube, which automatically graphs the equation. As the equation is evaluated (before it is graphed), “evaluation percentages” are shown in the upper-left part of the screen. Press <strong>F2</strong> 6.</td>
<td>![Graph Display]</td>
</tr>
</tbody>
</table>

**Note:** If you have already used 3D graphing, the graph may be shown in expanded view. When you animate the graph, the screen returns to normal view automatically. (Except for animation, you can do the same things in normal and expanded view.)

Press **×** (press **×** to switch between expanded and normal view).

5. Animate the graph by decreasing the eyeφ Window variable value. θ or ψ may affect eyeθ and eyeψ, but to a lesser extent than eyeφ.

To animate the graph continuously, press and hold the cursor for about 1 second and then release it. To stop, press **ENTER**.

Press **⊙** eight times.
<table>
<thead>
<tr>
<th>Steps and keystrokes</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>6. Return the graph to its initial orientation. Then move the viewing angle along the “viewing orbit” around the graph.</td>
<td>![Initial graph orientation]</td>
</tr>
<tr>
<td>Press 0 (zero, not the letter O) <strong>0 0 0</strong></td>
<td>![Graph orbit]</td>
</tr>
<tr>
<td>7. View the graph along the x axis, the y axis, and then the z axis.</td>
<td>![Graph x-axis]</td>
</tr>
<tr>
<td>Press X</td>
<td>![Graph y-axis]</td>
</tr>
<tr>
<td>This graph has the same shape along the y axis and x axis.</td>
<td>![Graph z-axis]</td>
</tr>
<tr>
<td>Press Y</td>
<td>![Graph x-axis]</td>
</tr>
<tr>
<td>Press Z</td>
<td>![Graph x-axis]</td>
</tr>
<tr>
<td>8. Return to the initial orientation.</td>
<td>![Initial graph orientation]</td>
</tr>
<tr>
<td>Press 0 (zero)</td>
<td>![Graph orbit]</td>
</tr>
<tr>
<td>Steps and keystrokes</td>
<td>Display</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------</td>
</tr>
<tr>
<td>9. Display the graph in different graph format styles.</td>
<td><img src="image1" alt="Hidden Surface" /></td>
</tr>
<tr>
<td>Press [1] (press [1] to switch from each style to the next)</td>
<td><img src="image2" alt="Contour Levels" /></td>
</tr>
<tr>
<td></td>
<td><img src="image3" alt="Wire and Contour" /></td>
</tr>
<tr>
<td></td>
<td><img src="image4" alt="Wire Frame" /></td>
</tr>
</tbody>
</table>
Note: You can also display the graph as an implicit plot by using the **GRAPH FORMATS** dialog box ( [ ]). If you press [ ] to switch between styles, the implicit plot is not displayed.

**Differential Equation Graphing**

Graph the solution to the logistic 1st-order differential equation \( y' = 0.001y(100-y) \). Start by drawing only the slope field. Then enter initial conditions in the **Y= Editor** and interactively from the Graph screen.

**Steps and keystrokes**

1. Display the **MODE** dialog box. For **Graph** mode, select **DIFF EQUATIONS**.
   
   Press **MODE** [ ] 6  **ENTER**
2. Display and clear the Y= Editor. Then define the 1st-order differential equation:

\[ y_1'(t) = 0.001y_1(100 - y_1) \]

Press \( \times \) to enter the \( \times \) shown above. Do not use implied multiplication between the variable and parentheses. If you do, it is treated as a function call.

Leave the initial condition \( y_{i1} \) blank.

**Note:** With \( y_1' \) selected, the device will graph the \( y_1 \) solution curve, not the derivative \( y_1' \).

Press \( \downarrow \) [Y=] F1 8 ENTER ENTER .001 Y1 \( \times \) [ ] 100 \( - \) Y1 [ ] ENTER

3. Display the GRAPH FORMATS dialog box. Then set Axes = ON, Labels = ON, Solution Method = RK, and Fields = SLPFLD.

**Note:** To graph one differential equation, Fields must be set to SLPFLD or FLDOFF. If Fields=DIRFLD, an error occurs when you graph.

Press \( \downarrow \) \( \blacksquare \) \( \leftarrow \) \( \rightarrow \) 2 \( \rightarrow \) \( \rightarrow \) 2 \( \rightarrow \) \( \rightarrow \) 1 \( \rightarrow \) \( \rightarrow \) \( \rightarrow \) 1 [ENTER]
<table>
<thead>
<tr>
<th>Steps and keystrokes</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Display the <strong>Window Editor</strong>, and set the Window variables as shown to the right.</td>
<td></td>
</tr>
<tr>
<td>t₀=0, ( t_{\text{max}}=10 ), ( t_{\text{step}}=0.1 ), ( t_{\text{plot}}=0 ), ( x_{\text{min}}=-10 ), ( x_{\text{max}}=110 ), ( x_{\text{sc1}}=10 ), ( y_{\text{min}}=-10 ), ( y_{\text{max}}=120 ), ( y_{\text{sc1}}=10 ), ( n_{\text{curves}}=0 ), ( df_{\text{tol}}=0.001 ), ( f_{\text{ldres}}=20 )</td>
<td></td>
</tr>
<tr>
<td>Press the <strong>[WINDOW]</strong> key followed by the values shown.</td>
<td></td>
</tr>
<tr>
<td>5. Display the Graph screen.</td>
<td></td>
</tr>
<tr>
<td>Because you did not specify an initial condition, only the slope field is drawn (as specified by <strong>Fields=SLPFLD</strong> in the <strong>GRAPH FORMATS</strong> dialog box).</td>
<td></td>
</tr>
<tr>
<td>Press the <strong>[GRAPH]</strong> key.</td>
<td></td>
</tr>
<tr>
<td>6. Return to the <strong>Y= Editor</strong> and enter an initial condition:</td>
<td></td>
</tr>
<tr>
<td>( y_{i1}=10 )</td>
<td></td>
</tr>
<tr>
<td>Press the <strong>[Y=]</strong> key followed by <strong>ENTER</strong> and enter 10.</td>
<td></td>
</tr>
<tr>
<td>7. Return to the Graph screen.</td>
<td></td>
</tr>
<tr>
<td>Initial conditions entered in the <strong>Y= Editor</strong> always occur at ( t_{0} ). The graph begins at the initial condition and plots to the right. Then it plots to the left.</td>
<td></td>
</tr>
<tr>
<td>The initial condition is marked with a circle.</td>
<td></td>
</tr>
</tbody>
</table>
8. Return to the **Y= Editor** and change \( y_{i1} \) to enter two initial conditions as a list:

\[
y_{i1} = \{10, 20\}
\]

Press \( \text{[Y-]} \) \( \text{ENTER} \) \( 2\text{nd} \) \( \{ \} \) 10 \( , \) 20 \( 2\text{nd} \) \( \} \) \( \text{ENTER} \)

9. Return to the Graph screen.

Press \( \text{[GRAPH]} \)
10. To select an initial condition interactively, press: 

2nd [F8] 

When prompted, enter t=40 and y₁=45.

When selecting an initial condition interactively, you can specify a value for t other than the t₀ value entered in the 
**Y= Editor or Window Editor.**

Instead of entering t and y₁ after pressing 

2nd [F8] 

you can move the cursor to a point on the screen and then press **ENTER**.

You can use **F3** to trace curves for initial conditions specified in the **Y= Editor.** However, you cannot trace the curve for an initial condition selected interactively.

Press **2nd [F8] 40 ENTER 45 ENTER**
Additional Graphing Topics

From the Home screen, graph the piecewise defined function: \( y = -x \) when \( x < 0 \) and \( y = 5 \cos(x) \) when \( x \geq 0 \). Draw a horizontal line across the top of the cosine curve. Then save a picture of the displayed graph.

Steps and keystrokes

1. Display the **MODE** dialog box. For **Graph** mode, select **FUNCTION**. For **Angle** mode, select **RADIANS**.

   Press **MODE** \( \uparrow \downarrow \downarrow \downarrow \downarrow \uparrow 1 \) **ENTER**

2. Display the Home screen. Use the **Graph** command and the **when** function to specify the piecewise defined function.

   \( \text{F4} \) \( 2 \) selects **Graph** from the **Other** toolbar menu and automatically adds a space.

   Press **HOME** \( \text{F4} \) \( 2 \) \( \text{2nd} \) [a-lock] **WHEN** \( \text{alpha} \)

   \( ( \) \( X \) \( \text{2nd} \) [\( \text{<} \) 0 \( , \) \( (-) \) \( X \) \( , \) \( 5 \times \) \( \text{2nd} \) [\( \text{COS} \) \( X \)

   \( ) \) **ENTER**

3. Execute the **Graph** command, which automatically displays the Graph screen.

   The graph uses the current Window variables, which are assumed to be their standard values (\( \text{F2} \) 6) for this example.

   Press **ENTER**
<table>
<thead>
<tr>
<th>Steps and keystrokes</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Draw a horizontal line across the top of the cosine curve.</td>
<td>![Display Image with horizontal line]</td>
</tr>
<tr>
<td>The calculator remains in “horizontal” mode until you select a different operation or press [ESC].</td>
<td>![Display Image with horizontal line]</td>
</tr>
<tr>
<td>Press 2nd [F7] 5 ⋄ (until the line is positioned) ENTER</td>
<td>![Display Image with horizontal line]</td>
</tr>
<tr>
<td>5. Save a picture of the graph. Use PIC1 as the variable name for the picture.</td>
<td>![Display Image with save copy as]</td>
</tr>
<tr>
<td>Be sure to set Type = Picture. By default, it is set to GDB.</td>
<td>![Display Image with save copy as]</td>
</tr>
<tr>
<td>Press F1 2 2 PIC alpha 1 ENTER ENTER</td>
<td>![Display Image with save copy as]</td>
</tr>
<tr>
<td>6. Clear the drawn horizontal line.</td>
<td>![Display Image with horizontal line]</td>
</tr>
<tr>
<td>You can also press F4 to regraph.</td>
<td>![Display Image with horizontal line]</td>
</tr>
<tr>
<td>Press 2nd [F6] 1</td>
<td>![Display Image with horizontal line]</td>
</tr>
</tbody>
</table>
7. Open the saved picture variable to redisplay the graph with the line.

Be sure to set **Type = Picture**. By default, it is set to **GDB**.

Press \[F_1\] 1 \(\rightarrow\) 2 (if not already shown, also set Variable = pic1) \[ENTER\]

---

### Tables

Evaluate the function \(y = x^3 - 2x\) at each integer between -10 and 10. How many sign changes are there, and where do they occur?

---

### Steps and keystrokes

1. Display the **MODE** dialog box. For the **Graph** mode, select **FUNCTION**.

   Press **MODE** \(\rightarrow\) 1 \[ENTER\]
2. Display and clear the **Y= Editor**. Then define \( y_1(x) = x^3 - 2x \).

   Press \( \mathbf{Y=} \) \( \mathbf{F}1 \) 8 \( \mathbf{ENTER} \) \( \mathbf{ENTER} \) \( X \uparrow 3 \) 2 \( \mathbf{X} \) \( \mathbf{ENTER} \)

3. Set the table parameters to:

   - \( \text{tblStart} = -10 \)
   - \( \Delta \text{tbl} = 1 \)
   - \( \text{Graph} \rightarrow \text{Table} = \text{OFF} \)
   - \( \text{Independent} = \text{AUTO} \)

   Press \( \mathbf{\downarrow} \) [\( \mathbf{TBLSET} \)] \( \mathbf{-} \) 10 \( \mathbf{\rightarrow} \) 1 \( \mathbf{\rightarrow} \) 1 \( \mathbf{\rightarrow} \) 1 \( \mathbf{\rightarrow} \) 1 \( \mathbf{\rightarrow} \) \( \mathbf{\rightarrow} \) \( \mathbf{\rightarrow} \) \( \mathbf{\rightarrow} \)

4. Display the Table screen.

   Press \( \mathbf{\downarrow} \) [\( \mathbf{TABLE} \)]

5. Scroll through the table. Notice that \( y_1 \) changes sign at \( x = -1, 1, \) and 2.

   To scroll one page at a time, use \( \mathbf{2nd} \) \( \mathbf{\downarrow} \) and \( \mathbf{2nd} \) \( \mathbf{\rightarrow} \).

   Press \( \mathbf{\rightarrow} \) and \( \mathbf{\rightarrow} \) as necessary.
### Steps and keystrokes

6. Zoom in on the sign change between $x = -2$ and $x = -1$ by changing the table parameters to:

- tblStart = -2
- Δtbl = .1

Press **F2** [ ] 2 [ ] 1 ENTER ENTER

---

### Split Screens

Split the screen to show the **Y= Editor** and the Graph screen. Explore the behavior of a polynomial as its coefficients change.

---

### Steps and keystrokes

1. Display the **MODE** dialog box.
   - For **Graph**, select **FUNCTION**.
   - For **Split Screen**, select **LEFT-RIGHT**.
   - For **Split 1 App**, select **Y= Editor**.
   - For **Split 2 App**, select **Graph**.

Press **MODE** 1 [ ] 2 [ ] 3 [ ] 4 [ ] ENTER
2. Clear the **Y= Editor** and turn off any stat data plots. Define \( y_1(x) = .1x^3 - 2x + 6 \).

A thick border around the **Y= Editor** indicates it is active. When active, its entry line goes all the way across the display.

Press \( F_1 \) 8 ENTER \( F_5 \) 5 ENTER \( .1 \times \wedge 3 \) \( 2 \times + 6 \) ENTER

3. Select the **ZoomStd** viewing window, which switches to the Graph screen and graphs the function.

The thick border is now around the Graph screen.

Press \( F_2 \) 6

4. Switch to the **Y= Editor** and edit \( y_1(x) \) to change \( .1x^3 \) to \( .5x^3 \).

\( 2\text{nd} \) [±±] is the second function of **APPS**. The thick border is around the **Y= Editor**.

Press \( 2\text{nd} \) [±±] ENTER 4 ENTER 5 ENTER
5. Switch to the Graph screen, which regraphs the edited function.
   The thick border is around the Graph screen.
   Press 2nd [→]

6. Switch to the Y= Editor and open the Window Editor in its place.
   Press 2nd [→] ◼ [WINDOW]

7. Open the Home screen and then exit to a full-sized Home screen.
   Press:
   2nd [QUIT] HOME
Use the **Data/Matrix Editor** to create a one-column list variable. Then add a second column of information. Notice that the list variable (which can have only one column) is automatically converted into a data variable (which can have multiple columns).

### Steps and keystrokes

1. Use **[APPS]** to display the **Data/Matrix Editor**. Create a new list variable named **TEMP**.

   Press 3 → 3 ← ← TEMP **[ENTER] [ENTER]**

2. Enter a column of numbers. Then move the cursor up one cell (just to see that a highlighted cell’s value is shown on the entry line).

   **LIST** is shown in the upper-left corner to indicate a list variable.

   You can use ← instead of **[ENTER]** to enter information in a cell.

   Press 1 **[ENTER]** 2 **[ENTER]** 3 **[ENTER]** 4 **[ENTER]** 5 **[ENTER]** 6 **[ENTER]** ←
3. Move to column 2, and define its column header so that it is twice the value of column 1.

**DATA** is shown in the upper-left corner to indicate that the list variable was converted to a data variable.

Press \( \text{F4} \) 2 \( \times \) \( \text{alpha} \) C 1 ENTER

<table>
<thead>
<tr>
<th>DATA</th>
<th>Cell Header</th>
<th>Calc</th>
<th>Util</th>
<th>Stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>c2</td>
<td>c3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( b6c2=12 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: If you don’t need to save the current variable, use it as a *scratchpad*. The next time you need a variable for temporary data, clear the current variable and re-use it. This lets you enter temporary data without wasting memory by creating a new variable each time.

4. Move to the column 2 header cell to show its definition in the entry line.

When the cursor is on the header cell, you do not need to press \( \text{F4} \) to define it. Simply begin typing the expression.

Press 2nd \( \text{enter} \) \( \text{enter} \)

5. Clear the contents of the variable.

Simply clearing the data does not convert the data variable back into a list variable.

Press F1 8 ENTER
Based on a sample of seven cities, enter data that relates population to the number of buildings with more than 12 stories. Using Median-Median and linear regression calculations, find and plot equations to fit the data. For each regression equation, predict how many buildings of more than 12 stories you would expect in a city of 300,000 people.

**Steps and keystrokes**

1. Display the **MODE** dialog box. For **Graph** mode, select **FUNCTION**.
   
   Press **MODE** → 1 → ENTER

2. Use **APPS** to display the **Data/Matrix Editor**. Create a new data variable named **BUILD**.
   
   Press 3 ◄ ◄ BUILD ENTER ENTER
3. Using the sample data below, enter the population in column 1.

<table>
<thead>
<tr>
<th>Pop. (in 1000s)</th>
<th>Bldgs &gt; 12 stories</th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
<td>4</td>
</tr>
<tr>
<td>500</td>
<td>31</td>
</tr>
<tr>
<td>800</td>
<td>42</td>
</tr>
<tr>
<td>250</td>
<td>9</td>
</tr>
<tr>
<td>500</td>
<td>20</td>
</tr>
<tr>
<td>750</td>
<td>55</td>
</tr>
<tr>
<td>950</td>
<td>73</td>
</tr>
</tbody>
</table>

Press 150 ENTER 500 ENTER 800 ENTER
250 ENTER 500 ENTER 750 ENTER 950 ENTER

4. Move the cursor to row 1 in column 2 (r1c2). Then enter the corresponding number of buildings.

.moves the cursor to the top of the page. After typing data for a cell, you can press ENTER or to enter the data and move the cursor down one cell. Pressing enters the data and moves the cursor up one cell.

Press 4 ENTER 31 ENTER 42 ENTER
9 ENTER 20 ENTER 55 ENTER 73 ENTER
5. Move the cursor to row 1 in column 1 (r1c1). Sort the data in ascending order of population.

This sorts column 1 and then adjusts all other columns so that they retain the same order as column 1. This is critical for maintaining the relationships between columns of data.

To sort column 1, the cursor can be anywhere in column 1. This example has you press

so that you can see the first four rows.

Press ◄ ◄ ◄ [2nd] [F6] 4

6. Display the **Calculate** dialog box. Set

**Calculation Type = MedMed**

\( x = C1 \)

\( y = C2 \)

**Store RegEQ to = y1(x)**


7. Perform the calculation to display the MedMed regression equation.

As specified on the **Calculate** dialog box, this equation is stored in \( y1(x) \).

Press ENTER
### Steps and keystrokes

<table>
<thead>
<tr>
<th>Steps</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>8. Close the <strong>STAT VARS</strong> screen. The <strong>Data/Matrix Editor</strong> displays.</td>
<td></td>
</tr>
<tr>
<td>Press <strong>ENTER</strong></td>
<td></td>
</tr>
</tbody>
</table>
| 9. Display the **Calculate** dialog box. Set: **Calculation Type = LinReg**  
  \[ x = C1 \]  
  \[ y = C2 \]  
  **Store RegEQ to = y2(x)** | ![Image](image1.png)  
  Press **F5** |  |
| 10. Perform the calculation to display the **LinReg** regression equation.  
  This equation is stored in \[ y2(x) \]. | ![Image](image2.png)  
  Press **ENTER** |  |
| 11. Close the **STAT VARS** screen. The **Data/Matrix Editor** displays. |  |
| Press **ENTER** |  |
| 12. Display the Plot Setup screen.  
  **Plot 1** is highlighted by default.  
  **F3** lets you clear highlighted Plot settings. | ![Image](image3.png)  
  Press **F2** |  |
13. Define Plot 1 as:
   *Plot Type = Scatter*
   *Mark = Box*
   *x = C1*
   *y = C2*

   Notice the similarities between this and the **Calculate** dialog box.

   Press **F1** 1 ↘ 1 ↘ C **alpha** 1 ↘ **alpha** C2

14. Save the plot definition and return to the Plot Setup screen.

   Notice the shorthand notation for Plot 1’s definition.

   Press **ENTER** twice
15. Display the **Y= Editor**. For \( y_1(x) \), the MedMed regression equation, set the display style to **Dot**.

**Note:** Depending on the previous contents of your **Y= Editor**, you may need to move the cursor to \( y_1 \).

**PLOTS** 1 at the top of the screen means that **Plot 1** is selected.

Notice that \( y_1(x) \) and \( y_2(x) \) were selected when the regression equations were stored.

Press \( \downarrow \) [Y=] 2nd [F6] 2

16. Scroll up to highlight **Plot 1**.

The displayed shorthand definition is the same as on the Plot Setup screen.

Press \( \uparrow \)

17. Use **ZoomData** to graph **Plot 1** and the regression equations \( y_1(x) \) and \( y_2(x) \).

**ZoomData** examines the data for all selected stat plots and adjusts the viewing window to include all points.

Press \( \text{F2} \) 9
Steps and keystrokes

18. Return to the current session of the **Data/Matrix Editor**.

Press **APPS D ENTER ENTER**

19. Enter a title for column 3. Define column 3’s header as the values predicted by the MedMed line.

To enter a title, the cursor must highlight the title cell at the very top of the column.

[F4] lets you define a header from anywhere in a column. When the cursor is on a header cell, pressing [F4] is not required.

Press **ınd [a-lock] MED alpha ENTER F4 Y1 ( alpha C1 ) ENTER**

20. Enter a title for column 4. Define column 4’s header as the residuals (difference between observed and predicted values) for MedMed.

Press **ınd [a-lock] RESID alpha ENTER alpha C2 - alpha C3 ENTER**

21. Enter a title for column 5. Define column 5’s header as the values predicted by the LinReg line.

Press **ınd [a-lock] LIN alpha ENTER F4 Y2 ( alpha C1 ) ENTER**
### Steps and keystrokes

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>22.</strong> Enter a title for column 6. Define column 6’s header as the residuals for LinReg.</td>
<td><img src="chart1.png" alt="Display" /></td>
</tr>
<tr>
<td>Press <strong>Å</strong> <strong>Å</strong> <a href="chart1.png">2nd [a-lock] RESID alpha ENTER</a></td>
<td><img src="chart2.png" alt="Display" /></td>
</tr>
<tr>
<td><img src="chart3.png" alt="Display" /></td>
<td></td>
</tr>
<tr>
<td>Press <strong>F4</strong> <strong>alpha C2 — alpha C5 ENTER</strong></td>
<td><img src="chart4.png" alt="Display" /></td>
</tr>
<tr>
<td><img src="chart5.png" alt="Display" /></td>
<td></td>
</tr>
<tr>
<td><strong>23.</strong> Display the Plot Setup screen and deselect <strong>Plot 1</strong>.</td>
<td><img src="chart6.png" alt="Display" /></td>
</tr>
<tr>
<td>Press <strong>F2</strong> <strong>F4</strong></td>
<td><img src="chart7.png" alt="Display" /></td>
</tr>
<tr>
<td><img src="chart8.png" alt="Display" /></td>
<td></td>
</tr>
<tr>
<td><strong>24.</strong> Highlight <strong>Plot 2</strong> and define it as: <strong>Plot Type = Scatter</strong> <strong>Mark = Box</strong> <strong>x = C1</strong> <strong>y = C4</strong> (MedMed residuals)</td>
<td><img src="chart9.png" alt="Display" /></td>
</tr>
<tr>
<td>Press <strong>Å</strong> <strong>F1</strong> <strong>Å</strong> <strong>C alpha 1</strong> <strong>α</strong> <strong>α C4 ENTER ENTER</strong></td>
<td><img src="chart10.png" alt="Display" /></td>
</tr>
<tr>
<td><img src="chart11.png" alt="Display" /></td>
<td></td>
</tr>
<tr>
<td><strong>25.</strong> Highlight <strong>Plot 3</strong> and define it as: <strong>Plot Type = Scatter</strong> <strong>Mark = Plus</strong> <strong>x = C1</strong> <strong>y = C6</strong> (LinReg residuals)</td>
<td><img src="chart12.png" alt="Display" /></td>
</tr>
<tr>
<td>Press <strong>Å</strong> <strong>F1</strong> <strong>Å</strong> <strong>3</strong> <strong>C alpha 1</strong> <strong>α</strong> <strong>αC6 ENTER ENTER</strong></td>
<td><img src="chart13.png" alt="Display" /></td>
</tr>
<tr>
<td><img src="chart14.png" alt="Display" /></td>
<td></td>
</tr>
<tr>
<td>Steps and keystrokes</td>
<td>Display</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| 26. Display the **Y= Editor** and turn all the \( y(x) \) functions off.  
> From \( \text{F}5 \), select **3:Functions Off**, not **1:All Off**.  
> Plots 2 and 3 are still selected.  
> Press \( \downarrow \) [Y=] \( \text{F}5 \) 3 | ![Display](image) |
| 27. Use **ZoomData** to graph the residuals.  
> □ marks the MedMed residuals;  
> + marks the LinReg residuals.  
> Press \( \text{F}2 \) 9 | ![Display](image) |
> Press \( \text{HOME} \) | ![Display](image) |
| 29. Use the MedMed (\( y_1(x) \)) and LinReg (\( y_2(x) \)) regression equations to calculate values for \( x = 300 \) (300,000 population).  
> The **round** function (\( \text{2nd} \) [MATH] 1 3) ensures that results show an integer number of buildings.  
> After calculating the first result, edit the entry line to change \( y_1 \) to \( y_2 \).  
> Press \( \text{2nd} \) [MATH] 1 3 [Y1] (300 \( \downarrow \)) \( \downarrow \) 0 \( \downarrow \) ENTER \( \downarrow \) (eight times) \( \downarrow \) 2 ENTER | ![Display](image) |
Programming

Write a program that prompts the user to enter an integer, sums all integers from 1 to the entered integer, and displays the result.

Steps and keystrokes

1. Use APPS to display the Program Editor. 
   Create a new program. 
   Press 3

2. Type PROG1 (with no spaces) as the name of the new program variable. 
   Press ° ° PROG alpha 1

3. Display the “template” for a new program. 
   The program name, Prgm, and EndPrgm are shown automatically. 
   After typing in an input box such as Variable, you must press [ENTER] twice. 
   Press [ENTER] twice.
4. Type the following program lines.

<table>
<thead>
<tr>
<th>Steps and keystrokes</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request &quot;Enter an integer&quot;, n</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td>Displays a dialog box that prompts “Enter an integer”, waits for the user to enter a value, and stores it (as a string) to variable n.</td>
<td></td>
</tr>
<tr>
<td>[expr(n) \Rightarrow n]</td>
<td>Converts the string to a numeric expression.</td>
</tr>
<tr>
<td>[0 \Rightarrow temp]</td>
<td>Creates a variable named temp and initializes it to 0.</td>
</tr>
<tr>
<td>[For i,1,n,1]</td>
<td>Starts a For loop based on variable i. First time through the loop, i = 1. At end of loop, i is incremented by 1. Loop continues until i &gt; n.</td>
</tr>
<tr>
<td>[temp+i \Rightarrow temp]</td>
<td>Adds current value of i to temp.</td>
</tr>
<tr>
<td>EndFor</td>
<td>Marks the end of the For loop.</td>
</tr>
<tr>
<td>Disp temp</td>
<td>Displays the final value of temp.</td>
</tr>
</tbody>
</table>

Type the program lines as shown. Press [ENTER] at the end of each line.
5. Go to the Home screen. Enter the program name, followed by a set of parentheses.

You must include ( ) even when there are no arguments for the program.

The program displays a dialog box with the prompt specified in the program.

Press **HOME** 2nd [a-lock] PROG alpha 1 ( ) ENTER

6. Type 5 in the displayed dialog box.

Press 5

7. Continue with the program. The **Disp** command displays the result on the Program I/O screen.

The result is the sum of the integers from 1 through 5.

Although the Program I/O screen looks similar to the Home screen, it is for program input and output only. You cannot perform calculations on the Program I/O screen.

Press **ENTER** twice

Output from other programs may still be on the screen.
8. Leave the Program I/O screen and return to the Home screen.

You can also press ESC, 2nd [QUIT], or HOME to return to the Home screen.

Press F5

**Text Operations**

Start a new Text Editor session. Then practice using the Text Editor by typing whatever text you want. As you type, practice moving the text cursor and correcting any typos you may enter.

### Steps and keystrokes

<table>
<thead>
<tr>
<th>Steps and keystrokes</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.</strong> Start a new session of the Text Editor.</td>
<td></td>
</tr>
<tr>
<td>Press 3</td>
<td><img src="image1" alt="Text Editor" /></td>
</tr>
<tr>
<td><strong>2.</strong> Create a text variable called <strong>TEST</strong>, which will automatically store any text you enter in the new session.</td>
<td><img src="image2" alt="NEW" /></td>
</tr>
<tr>
<td>Use the <strong>MAIN</strong> folder, shown as the default on the <strong>NEW</strong> dialog box.</td>
<td></td>
</tr>
<tr>
<td>After typing in an input box such as <strong>Variable</strong>, you must press ENTER twice.</td>
<td></td>
</tr>
<tr>
<td>Press <strong>TEST</strong> ENTER ENTER</td>
<td></td>
</tr>
</tbody>
</table>
3. Type some sample text.
   - To type a single uppercase letter, press \( \text{up} \) and then the letter.
   - To type a space, press \( \text{alpha} [\text{-}] \) (alpha function of the \( [-] \) key).
   - To type a period, press \( \text{alpha} \) to turn alpha-lock off, press \( \text{left} \), and then press \( \text{2nd} [\text{a-lock}] \) to turn alpha-lock on again.

Practice editing your text by using:
   - The cursor pad to move the text cursor.
   - \( \text{left} \) or \( \text{up} \) [\text{DEL}] to delete the character to the left or right of the cursor, respectively.

Press \( \text{2nd} [\text{a-lock}] \) and type anything you want.

4. Leave the **Text Editor** and display the Home screen.

Your text session was stored automatically as you typed. Therefore, you do not need to save the session manually before exiting the **Text Editor**.

Press \( \text{HOME} \)
Consider the equation \( a = \frac{m_2 - m_1}{m_2 + m_1} \times g \), where the known values are \( m_2 = 10 \) and \( g = 9.8 \). If you assume that \( a = \frac{1}{3} g \), find the value of \( m_1 \).

Steps and keystrokes  Display

5. Return to the current session on the **Text Editor**. Notice that the displayed session is exactly the same as you left it.

Press 2nd [→]

---

**Numeric Solver**

Consider the equation \( a = \frac{m_2 - m_1}{m_2 + m_1} \times g \), where the known values are \( m_2 = 10 \) and \( g = 9.8 \). If you assume that \( a = \frac{1}{3} g \), find the value of \( m_1 \).

Steps and keystrokes  Display

1. Use APPS to display the **Numeric Solver**.

   ![Numeric Solver Display](image)

2. Enter the equation.

   When you press ENTER or ↓, the screen lists the variables used in the equation.

   Press  alpha A  \( \equiv \) \( ( \)  alpha M2  \( - \)  alpha M1  \( ) \) \( \div \) \( ( \)  alpha M2  \( + \)  alpha M1  \( ) \) \( \times \)  alpha G  ENTER

   ![Equation Input](image)
### Steps and keystrokes

<table>
<thead>
<tr>
<th>Steps</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Enter values for each variable, except the unknown variable m1.</td>
<td>![Display Image]</td>
</tr>
<tr>
<td>Define m2 and g first. Then define a. (You must define g before you can define a in terms of g.) Accept the default for bound. If a variable has been defined previously, its value is shown as a default.</td>
<td>![Display Image]</td>
</tr>
<tr>
<td>Press ( \Rightarrow 10 \Rightarrow 9.8 \Rightarrow ) ( \alpha G \div 3 )</td>
<td></td>
</tr>
<tr>
<td>4. Move the cursor to the unknown variable m1.</td>
<td>![Display Image]</td>
</tr>
<tr>
<td>Optionally, you can enter an initial guess for m1. Even if you enter a value for all variables, the Numeric Solver solves for the variable marked by the cursor.</td>
<td>![Display Image]</td>
</tr>
<tr>
<td>Press ( \Rightarrow \Rightarrow )</td>
<td>g/3 is evaluated when you move the cursor off the line.</td>
</tr>
<tr>
<td>5. Solve for the unknown variable.</td>
<td>![Display Image]</td>
</tr>
<tr>
<td>To check the solution’s accuracy, the left and right sides of the equation are evaluated separately. The difference is shown as left-rt. If the solution is precise, left-rt=0.</td>
<td>![Display Image]</td>
</tr>
<tr>
<td>Press F2</td>
<td></td>
</tr>
</tbody>
</table>

\( \alpha \) marks the calculated values.
6. Graph the solution using a **ZoomStd** viewing window.

The graph is displayed in a split screen. You can explore the graph by tracing, zooming, etc.

The variable marked by the cursor (unknown variable m1) is on the x axis, and left-rt is on the y axis.

Press **F3 3**

7. Return to the **Numeric Solver** and exit the split screen.

You can press **ENTER** or **Ω** to redisplay the list of variables.

Press **2nd [+] F3 2**
Calculate 10 binary (base 2) + F hexadecimal (base 16) + 10 decimal (base 10). Then, use the operator to convert an integer from one base to another. Finally, see how changing the Base mode affects the displayed results.

**Steps and keystrokes**

1. Display the MODE dialog box, Page 2. For Base mode, select DEC as the default number base.

   Integer results are displayed according to the Base mode. Fractional and floating-point results are always displayed in decimal form.

   Press $\text{MODE} \; F2$ (use $\leftarrow$ to move to Base mode) $\Rightarrow$ 1 $\text{ENTER}$

2. Calculate 0b10+0hF+10.

   To enter a binary or hex number, you must use the 0b or 0h prefix (zero and the letter B or H). Otherwise, the entry is treated as a decimal number.

   **Note:** The 0b or 0h prefix is a zero, not the letter O, followed by B or H.

   Press $0 \; \text{alpha} \; B \; 10 + 0 \; \text{2nd} \; [a\text{-lock}] \; \text{HF}$ $\alpha + 10 \; \text{ENTER}$
Steps and keystrokes | Display
---|---
3. Add 1 to the result and convert it to binary.
- \(2^n [\uparrow]\) displays the \(\uparrow\) conversion operator.
- Press \(+ 1 \ 2^n [\uparrow] \ 2^n [\text{a-lock}] \) BIN alpha ENTER

4. Add 1 to the result and convert it to hexadecimal.
- Press \(+ 1 \ 2^n [\uparrow] \ 2^n [\text{a-lock}] \) HEX alpha ENTER

5. Add 1 to the result and leave it in the default decimal base.
- Results use the 0b or 0h prefix to identify the base.
- Press \(+ 1 \) ENTER

6. Change the Base mode to HEX.
- When Base = HEX or BIN, the magnitude of a result is restricted to certain size limitations.
- Press \(\text{MODE} \ F2\) (use \(\odot\) to move to Base mode) \(\odot\) 2 ENTER

7. Calculate 0b10+0hF+10.
- Press \(0 \text{alpha} \ B \ 10 + 0 \ 2^n [\text{a-lock}] \) HF alpha + 10 ENTER
Memory and Variable Management

Assign values to a variety of variable data types. Use the VAR-LINK screen to view a list of the defined variables. Then move a variable to the user data archive memory and explore the ways in which you can and cannot access an archived variable. (Archived
variables are locked automatically.) Finally, unarchive the variable and delete the unused variables so that they will not take up memory.

**Steps and keystrokes**

1. From the Home screen, assign variables with the following variable types.

   Expression: \( 5 \rightarrow x_1 \)

   Function: \( x^2 + 4 \rightarrow f(x) \)

   List: \( \{5,10\} \rightarrow L_1 \)

   Matrix: \([30,25]\) \(\rightarrow m_1\)

   Press `HOME`, `CLEAR`, `5` `STO→` `X_1` `ENTER` `X^2` `4` `STO→` `ALPHA` `F_2` `ENTER` `2ND` `[1]` `5` `10` `2ND` `[,]` `STO→` `ALPHA` `L_1` `ENTER` `2ND` `[1]` `30` `,` `25` `2ND` `[,]` `STO→` `ALPHA` `M_1` `ENTER`

2. Suppose you start to perform an operation using a function variable but can’t remember its name.

   Press `5` `×`

3. Display the VAR-LINK screen.

   This example assumes that the variables assigned above are the only ones defined.

   Press `2ND` `VAR-LINK`
4. Change the screen’s view to show only function variables.

Although this may not seem particularly useful in an example with four variables, consider how useful it could be if there were many variables of all different types.

Press \( \text{F2} \) \( \text{ENTER} \)

5. Highlight the \( f \) function variable, and view its contents.

Notice that the function was assigned using \( f(x) \) but is listed as \( f \) on the screen.

Press \( \text{2nd} \) \( [F6] \)

6. Close the Contents window.

Press \( \text{ESC} \)

7. With the \( f \) variable still highlighted, close \( \text{VAR-LINK} \) to paste the contents of the variable to the entry line. Notice that “(” is pasted.

Press \( \text{ENTER} \)

8. Complete the operation.

Press \( 2 \) \( \text{ENTER} \)
Archiving a variable

Steps and keystrokes

1. Redisplay **VAR-LINK**, and highlight the variable you want to archive.

   The previous change in view is no longer in effect. The screen lists all defined variables.

   Press **2nd [VAR-LINK]** (use **Cursor** to highlight x1)

2. Use the **F1 Manage** toolbar menu to archive the variable.

   • indicates the variable is archived.

   Press **F1 8**

3. Return to the Home screen and use the archived variable in a calculation.

   Press **HOME 6 × X1 ENTER**
### Steps and keystrokes

<table>
<thead>
<tr>
<th>Steps and keystrokes</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Attempt to store a different value to the archived variable.</td>
<td><img src="image" alt="Error Message" /></td>
</tr>
<tr>
<td>Press 10 <strong>STO</strong> X1 ENTER</td>
<td></td>
</tr>
<tr>
<td>5. Cancel the error message.</td>
<td><img src="image" alt="Error Message" /></td>
</tr>
<tr>
<td>Press ESC</td>
<td></td>
</tr>
<tr>
<td>6. Use <strong>VAR-LINK</strong> to unarchive the variable.</td>
<td><img src="image" alt="Variable Unarchive" /></td>
</tr>
<tr>
<td>Press 2nd <strong>[VAR-LINK]</strong> (use ▲ to highlight x1) F1 9</td>
<td></td>
</tr>
<tr>
<td>7. Return to the Home screen and store a different value to the unarchived variable.</td>
<td><img src="image" alt="Home Screen" /></td>
</tr>
<tr>
<td>Press HOME ENTER</td>
<td></td>
</tr>
</tbody>
</table>
Deleting variables

Steps and keystrokes

1. Display VAR-LINK, and use the F5 All toolbar menu to select all variables.

A ✓ mark indicates items that are selected. Notice that this also selected the MAIN folder.

Note: Instead of using F5 (if you don’t want to delete all your variables), you can select individual variables. Highlight each variable to delete and press F4.

Press F5 1

2. Use F1 to delete.

Note: You can press ← (instead of F1 1) to delete the marked variables.

Press F1 1

3. Confirm the deletion.

Press ENTER
4. Because [F5] 1 also selected the MAIN folder, an error message states that you cannot delete the MAIN folder. Acknowledge the message.

When VAR-LINK is redisplayed, the deleted variables are not listed.

Press [ENTER]

5. Close VAR-LINK and return to the current application (Home screen in this example).

When you use [ESC] (instead of [ENTER]) to close VAR-LINK, the highlighted name is not pasted to the entry line.

Press [ESC]
Operating the Calculator

Turning the Calculator On and Off

You can turn your graphing calculator on and off manually by using the ON and 2nd OFF (or [OFF]) keys. To prolong battery life, the APD™ (Automatic Power Down™) feature lets the calculator turn itself off automatically.

Turning the Calculator On

Press ON.

- If you turned the unit off by pressing 2nd OFF, the unit returns to either the Apps desktop or the Home screen.
- If you turned the unit off by pressing [OFF] or if the unit turned itself off through APD, the unit returns to whichever application you used last.
**Turning the Calculator Off**

You can use either of the following keys to turn off your graphing calculator.

<table>
<thead>
<tr>
<th>Press:</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2nd [OFF] (press 2nd and then press [OFF]) | Settings and memory contents are retained by the Constant Memory™ feature. However:  
  - You cannot use 2nd [OFF] if an error message is displayed.  
  - When you turn the calculator on again, it displays either the Home screen or the Apps desktop (regardless of the last application you used). |
| [OFF] (press and then press [OFF]) | Similar to 2nd [OFF] except:  
  - You can use [OFF] if an error message is displayed.  
  - When you turn the calculator on again, it will be exactly as you left it. |

**Note:** [OFF] is the second function of the ON key.

**APD (Automatic Power Down)**

After several minutes without any activity, the calculator turns itself off automatically. This feature is called APD.

When you press ON, the calculator will be exactly as you left it.

- The display, cursor, and any error conditions are exactly as you left them.
- All settings and memory contents are retained.
APD does not occur if a calculation or program is in progress, unless the program is paused. If a program is running, but waiting for a key press, APD will occur after several minutes of inactivity.

**Setting the Display Contrast**

The brightness and contrast of the display depend on room lighting, battery freshness, viewing angle, and the adjustment of the display contrast. The contrast setting is retained in memory when the graphing calculator is turned off.

**Adjusting the Display Contrast**

You can adjust the display contrast to suit your viewing angle and lighting conditions.

<table>
<thead>
<tr>
<th>To:</th>
<th>Press and hold both:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decrease (lighten) the contrast</td>
<td>♦ and −</td>
</tr>
<tr>
<td>Increase (darken) the contrast</td>
<td>♦ and +</td>
</tr>
</tbody>
</table>

*Contrast keys*

If you press and hold ♦ − or ♦ + too long, the display may go completely black or blank. To make finer adjustments, hold ♦ and then tap − or +.
When to Replace Batteries

As the batteries get low, the display begins to dim (especially during calculations) and you must increase the contrast. If you have to increase the contrast frequently, replace the four alkaline batteries.

**Note:** The display may be very dark after you change batteries. Use ☼ to lighten the display.

The status line along the bottom of the display also gives battery information.

<table>
<thead>
<tr>
<th>Indicator in status line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BATT</td>
<td>Batteries are low.</td>
</tr>
<tr>
<td>BATT</td>
<td>Replace batteries as soon as possible.</td>
</tr>
</tbody>
</table>

The TI-89 Titanium Keyboard

Most keys can perform two or more functions, depending on whether you first press a modifier key.
1  [F1] – [2nd] [F8] open toolbar menus. Select applications (when used with [▼])
2  [2nd], [▼], [▲], and [alpha] add functionality by increasing the available key commands.
3  X, Y, and Z are often used in symbolic calculations.

4  [◄], [►], [▲], and [▼] move the cursor.
5  [APPs] lets you select an application.
6  [ENTER] evaluates an expression, executes an instruction, selects a menu item, etc.
## Modifier Keys

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2nd] (second)</td>
<td>Accesses the second function of the next key you press. On the keyboard, these are printed in the same color as the [2nd] key.</td>
</tr>
<tr>
<td>[\downarrow] (diamond)</td>
<td>Activates keys that select certain applications, menu items, and other operations from the keyboard. On the keyboard, these are printed in the same color as the [\downarrow] key.</td>
</tr>
<tr>
<td>[\uparrow] (shift)</td>
<td>Types an uppercase character for the next letter key you press. [\uparrow] is also used with [\leftarrow] and [\rightarrow] to highlight characters in the entry line for editing purposes.</td>
</tr>
<tr>
<td>[alpha]</td>
<td>Used to type alphabetic letters, including a space character. On the keyboard, these are printed in the same color as the [alpha] key.</td>
</tr>
</tbody>
</table>

**Note:** Information is available about using \[\uparrow\] and \[alpha\].

### Examples of [2nd] and [diamond] Modifiers

The \[\text{ESC}\] key is one of several keys that can perform three operations, depending on whether you first press \[2nd\] or \[\downarrow\].

The following TI-89 Titanium example shows using the \[2nd\] or \[\downarrow\] modifier key with the \[\text{ESC}\] key.
Some keys perform only one additional operation, which may require either \texttt{2nd} or \texttt{\textdagger}, depending on the color in which the operation is printed on the keyboard and where it is positioned above the key.

When you press a modifier such as \texttt{2nd} or \texttt{\textdagger}, a \texttt{2ND} or \texttt{\textdagger} indicator appears in the status line at the bottom of the display. If you press a modifier by accident, press it again (or press \texttt{ESC}) to cancel its effect.

**Other Important Keys You Need to Be Familiar With**

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{\textdagger} [Y=]</td>
<td>Displays the Y= Editor.</td>
</tr>
<tr>
<td>\texttt{\textdagger} [WINDOW]</td>
<td>Displays the Window Editor.</td>
</tr>
<tr>
<td>\texttt{\textdagger} [GRAPH]</td>
<td>Displays the Graph screen.</td>
</tr>
<tr>
<td>\texttt{\textdagger} [TBLSET]</td>
<td>Sets parameters for the Table screen.</td>
</tr>
<tr>
<td>Key</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
</tr>
<tr>
<td>◆ [TABLE]</td>
<td>Displays the Table screen.</td>
</tr>
<tr>
<td>◆ [CUT]</td>
<td>These keys let you edit entered information by performing a cut, copy, or paste operation.</td>
</tr>
<tr>
<td>◆ [COPY]</td>
<td></td>
</tr>
<tr>
<td>◆ [PASTE]</td>
<td></td>
</tr>
<tr>
<td>2nd [←±]</td>
<td>Toggles between the last two chosen Apps or between split screen portions.</td>
</tr>
<tr>
<td>2nd [CUSTOM]</td>
<td>Toggles the custom menu on and off.</td>
</tr>
<tr>
<td>2nd [▼]</td>
<td>Converts measurement units.</td>
</tr>
<tr>
<td>2nd [–]</td>
<td>Designates a measurement unit.</td>
</tr>
<tr>
<td>←</td>
<td>Deletes the character to the left of the cursor (backspaces).</td>
</tr>
<tr>
<td>2nd [INS]</td>
<td>Toggles between insert and overtype mode for entering information.</td>
</tr>
<tr>
<td>◆ [DEL]</td>
<td>Deletes the character to the right of the cursor.</td>
</tr>
<tr>
<td>◎</td>
<td>Enters the &quot;with&quot; operator, which is used in symbolic calculations.</td>
</tr>
<tr>
<td>2nd [∫], 2nd [d]</td>
<td>Performs integrations and derivatives.</td>
</tr>
<tr>
<td>2nd [∠]</td>
<td>Designates an angle in polar, cylindrical, and spherical coordinates.</td>
</tr>
<tr>
<td>2nd [MATH]</td>
<td>Displays the MATH menu.</td>
</tr>
</tbody>
</table>
Entering Alphabetic Characters

Alphabetic characters are used in expressions such as $x^2+y^2$ to enter variable names and in the Text Editor (Text Editor module).

Entering a Letter Character on the TI-89 Titanium

The letters x, y, z, and t are commonly used in algebraic expressions. So that you can type them quickly, these letters are primary keys on the TI-89 Titanium keyboard.

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd [MEM]</td>
<td>Displays the MEMORY screen.</td>
</tr>
<tr>
<td>2nd [VAR-LINK]</td>
<td>Displays the VAR-LINK screen for managing variables and Flash applications.</td>
</tr>
<tr>
<td>2nd [RCL]</td>
<td>Recalls the contents of a variable.</td>
</tr>
<tr>
<td>2nd [UNITS]</td>
<td>Displays the UNITS dialog box.</td>
</tr>
<tr>
<td>2nd [CHAR]</td>
<td>Displays the CHAR menu, which lets you select Greek letters, international accented characters, etc.</td>
</tr>
<tr>
<td>2nd [ENTRY], 2nd [ANS]</td>
<td>Recalls the previous entry and the last answer, respectively.</td>
</tr>
</tbody>
</table>
Other letters are available as the \texttt{alpha} function of another key, similar to the \texttt{2nd} and \texttt{\#} modifiers described in the previous section. For example:

\begin{itemize}
  \item \texttt{2nd} \texttt{[\.']} types \('', which is the same color as the \texttt{2nd} key.
  \item \texttt{[\#]} \texttt{A} displays an \texttt{A}, which is the same color as the \texttt{alpha} key.
\end{itemize}

**Typing Alphabetic Characters on the TI-89 Titanium**

<table>
<thead>
<tr>
<th>To:</th>
<th>Press:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type a single lowercase alpha character.</td>
<td>\texttt{alpha} and then the letter key (status line shows \texttt{[a-lock]})</td>
</tr>
<tr>
<td>Type a single uppercase alpha character.</td>
<td>\texttt{#} and then the letter key (status line shows \texttt{[alpha-lock]})</td>
</tr>
<tr>
<td>Type a space.</td>
<td>\texttt{alpha} \texttt{[#]} (alpha function of the \texttt{[#]} key)</td>
</tr>
<tr>
<td>Turn on lowercase alpha-lock.</td>
<td>\texttt{2nd} \texttt{[a-lock]} (status line shows \texttt{[a-lock]})</td>
</tr>
<tr>
<td>Turn on uppercase ALPHA-lock.</td>
<td>\texttt{#} \texttt{[a-lock]} (status line shows \texttt{[alpha-lock]})</td>
</tr>
<tr>
<td>Turn off alpha-lock.</td>
<td>\texttt{alpha} (turns off upper- and lowercase lock)</td>
</tr>
</tbody>
</table>

**Notes:**
• On the TI-89 Titanium, you do not need \texttt{alpha} or alpha-lock to type \(x, y, z,\) or \(t.\) But you must use \(1\) or uppercase ALPHA-lock for \(X, Y, Z,\) or \(T.\)

• On the TI-89 Titanium, alpha-lock is always turned off when you change applications, such as going from the Text Editor to the Home screen.

On the TI-89 Titanium, while either type of alpha-lock is on:

• To type a period, comma, or other character that is the primary function of a key, you must turn alpha-lock off.

• To type a second function character such as \(2\texttt{nd}\) \(\{\},\) you do not need to turn alpha-lock off. After you type the character, alpha-lock remains on.

**Automatic Alpha-Lock in TI-89 Titanium Dialog Boxes**

There are certain times when you do not need to press \texttt{alpha} or \texttt{2nd} to type alphabetic characters on the TI-89 Titanium. Automatic alpha-lock is turned on whenever a dialog box is first displayed. The automatic alpha-lock feature applies to these dialog boxes:

<table>
<thead>
<tr>
<th>Dialog box</th>
<th>Alpha-lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalog dialog box</td>
<td>All commands are listed in alphabetical order. Press a letter to go to the first command that begins with that letter.</td>
</tr>
<tr>
<td>Units dialog box</td>
<td>In each unit category, type the first letter of a unit or constant. See <em>Constants and Measurement Units</em> for more information.</td>
</tr>
<tr>
<td>Dialog boxes with entry fields</td>
<td>These include, but are not limited to: Create New Folder, Rename, and Save Copy As.</td>
</tr>
</tbody>
</table>
**Note:** To type a number, press [alpha] to turn alpha-lock off. Press [alpha] or [2nd] [a-lock] to resume typing letters.

Alpha-lock is *not* turned on in dialog boxes that require numeric-only entries. The dialog boxes that accept only numeric entries are: Resize Matrix, Zoom Factors, and Table Setup.

**For Special Characters**

Use the [2nd] [CHAR] menu to select from a variety of special characters. For more information, refer to “Entering Special Characters” in the *Text Editor* module.

**Entering Numbers**

The keypad lets you enter positive and negative numbers for your calculations. You can also enter numbers in scientific notation.

**Entering a Negative Number**

1. Press the negation key [(-)]. (Do not use the subtraction key [-].)
2. Type the number.

To see how your graphing calculator evaluates a negation in relation to other functions, refer to the Equation Operating System (EOS™) hierarchy in the *Technical Reference*.
module. For example, it is important to know that functions such as \( x^2 \) are evaluated before negation.

Use \( \text{□} \) and \( \text{□} \) to include parentheses if you have any doubt about how a negation will be evaluated.

<table>
<thead>
<tr>
<th>Evaluated as (-(2^2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-2^2)</td>
</tr>
<tr>
<td>((-2)^2)</td>
</tr>
<tr>
<td>((-2)^2)</td>
</tr>
</tbody>
</table>

If you use \( \text{□} \) instead of \( \text{□} \) (or vice versa), you may get an error message or you may get unexpected results. For example:

- \( 9 \text{ □} \ 7 = -63 \)
  - but –
  \( 9 \text{ □} \ 7 \) displays an error message.

- \( 6 \text{ □} \ 2 = 4 \)
  - but –
  \( 6 \text{ □} \ 2 = -12 \) since it is interpreted as \( 6(-2) \), implied multiplication.

- \( \text{□} \ 2 \text{ □} \ 4 = 2 \)
  - but –
  \( \text{□} \ 2 \text{ □} \ 4 \) subtracts 2 from the previous answer and then adds 4.

**Important:** Use \( \text{□} \) for subtraction and use \( \text{□} \) for negation.

**Entering a Number in Scientific Notation**

1. Type the part of the number that precedes the exponent. This value can be an expression.
2. Press: 

\[ \text{EE} \]

\( \varepsilon \) appears in the display.

3. Type the exponent as an integer with up to 3 digits. You can use a negative exponent.

Entering a number in scientific notation does not cause the answers to be displayed in scientific or engineering notation.

The display format is determined by the mode settings and the magnitude of the number.

\[ \begin{array}{cc}
\text{\( \approx 1.2345 \) } & \text{1.2345} \\
\text{123.45e-2} & \\
\text{FIND } & \text{RADIAN } \text{FUNC } 1.30
\end{array} \]

Represents \( 123.45 \times 10^{-2} \)

**Entering Expressions and Instructions**

You perform a calculation by evaluating an expression. You initiate an action by executing the appropriate instruction. Expressions are calculated and results are displayed according to the mode settings.
## Definitions

<table>
<thead>
<tr>
<th><strong>Expression</strong></th>
<th>Consists of numbers, variables, operators, functions, and their arguments that evaluate to a single answer. For example: ( \pi r^2 + 3 ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Enter an expression in the same order that it normally is written.</td>
</tr>
<tr>
<td>-</td>
<td>In most places where you are required to enter a value, you can enter an expression.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Operator</strong></th>
<th>Performs an operation such as +, −, *, ^.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Operators require an argument before and after the operator. For example: 4+5 and 5^2.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Function</strong></th>
<th>Returns a value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Functions require one or more arguments (enclosed in parentheses) after the function. For example: ( \sqrt{5} ) and ( \text{min}(5,8) ).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Instruction</strong></th>
<th>Initiates an action.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Instructions cannot be used in expressions.</td>
</tr>
<tr>
<td>-</td>
<td>Some instructions do not require an argument. For example: ClrHome.</td>
</tr>
<tr>
<td>-</td>
<td>Some require one or more arguments. For example: Circle 0,0,5.</td>
</tr>
</tbody>
</table>

**Note:** For instructions, do not put the arguments in parentheses.

## Notes:

- The *Technical Reference* module describes all of the built-in functions and instructions.
This guidebook uses the word command as a generic reference to both functions and instructions.

### Implied Multiplication

The graphing calculator recognizes implied multiplication, provided it does not conflict with a reserved notation.

<table>
<thead>
<tr>
<th>If you enter:</th>
<th>The calculator interprets it as:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Valid</strong></td>
<td></td>
</tr>
<tr>
<td>$2\pi$</td>
<td>$2*\pi$</td>
</tr>
<tr>
<td>$4\sin(46)$</td>
<td>$4*\sin(46)$</td>
</tr>
<tr>
<td>$5(1+2)$ or $(1+2)5$</td>
<td>$5*(1+2)$ or $(1+2)*5$</td>
</tr>
<tr>
<td>$[1,2]a$</td>
<td>$[a\ 2a]$</td>
</tr>
<tr>
<td>$2(a)$</td>
<td>$2*a$</td>
</tr>
<tr>
<td><strong>Invalid</strong></td>
<td></td>
</tr>
<tr>
<td>$xy$</td>
<td>Single variable named $xy$</td>
</tr>
<tr>
<td>$a(2)$</td>
<td>Function call</td>
</tr>
<tr>
<td>$a[1,2]$</td>
<td>Matrix index to element $a[1,2]$</td>
</tr>
</tbody>
</table>

### Parentheses

Expressions are evaluated according to the Equation Operating System (EOS™) hierarchy described in the *Technical Reference* module. To change the order of evaluation or just to ensure that an expression is evaluated in the order you require, use parentheses.

Calculations inside a pair of parentheses are completed first. For example, in $4(1+2)$, EOS first evaluates $(1+2)$ and then multiplies the answer by 4.
Entering an Expression

Type the expression, and then press [ENTER] to evaluate it. To enter a function or instruction name on the entry line, you can:

- Press its key, if available. For example, press: [2nd] [SIN]
  
  – or –

- Select it from a menu, if available. For example, select **2:abs** from the Number submenu of the MATH menu.

  – or –

- Type the name letter-by-letter from the keyboard. (On the TI-89 Titanium, use [alpha] and [2nd] [a-lock] to type letters.) You can use any mixture of uppercase or lowercase letters. For example, type **sin(** or **Sin(**.

Example

Calculate $3.76 \div (-7.9 + \sqrt{5}) + 2 \log 45$. Type the function name in this example.
### TI-89 Titanium

**Press** | **Display**
--- | ---
3.76 ÷ [( (-) 7.9 + | \[ \frac{3.76}{-7.9 + \sqrt{5}} \]
2nd [\(\sqrt{\)] | 2nd [\(\sqrt{\)] \text{ inserts } \sqrt{\text{ because its argument must be in parentheses.}}

5 2nd [\(\sqrt{\)}] | \[ \frac{3.76}{-7.9 + \sqrt{5}} \]
Use 2nd [\(\sqrt{\)}] once to close \(\sqrt{5}\) and again to close \((-7.9 + \sqrt{5})\).

2nd [a-lock] LOG | \[ \frac{3.76}{-7.9 + \sqrt{5}} \] + 2 \log(45)
alpha ( 45 ) | \[ \frac{3.76}{-7.9 + \sqrt{5}} \] \log requires ( ) around its argument.

---

**Note:** You can also select log by using [CATALOG](#)
Entering Multiple Expressions on a Line

To enter more than one expression or instruction at a time, separate them with a colon by pressing [2nd] [:].

1. Displays last result only.
2. ➔ is displayed when you press [STO➔] to store a value to a variable.

If an Entry or Answer Is Too Long for One Line

In the history area, if both the entry and its answer cannot be displayed on one line, the answer is displayed on the next line.

If an entry or answer is too long to fit on one line, ➔ is displayed at the end of the line.

To view the entire entry or answer:

1. Press [2nd] to move the cursor from the entry line up into the history area. This highlights the last answer.
2. As necessary, use and to highlight the entry or answer you want to view. For example, moves from answer to entry, up through the history area.

3. Use or or and to scroll right and left.

   **Note:** When you scroll to the right, is displayed at the beginning of the line.

4. To return to the entry line, press ESC.

**Continuing a Calculation**

When you press ENTER to evaluate an expression, the graphing calculator leaves the expression on the entry line and highlights it. You can continue to use the last answer or enter a new expression.

<table>
<thead>
<tr>
<th>If you press:</th>
<th>The calculator:</th>
</tr>
</thead>
<tbody>
<tr>
<td>+, −, ×, ÷,</td>
<td>Replaces the entry line with the variable ans(1), which lets you use the last answer as the beginning of another expression.</td>
</tr>
<tr>
<td>^, or STO↑</td>
<td>ERases the entry line and begins a new entry.</td>
</tr>
</tbody>
</table>

**Example**

Calculate 3.76 ÷ (−7.9 + √5). Then add 2 log 45 to the result.
Stopping a Calculation

When a calculation is in progress, BUSY appears on the right end of the status line. To stop the calculation, press ON.

There may be a delay before the Break message is displayed.

Press ESC to return to the current application.

Formats of Displayed Results

A result may be calculated and displayed in any of several formats. This section describes the modes and their settings that affect the display formats. You can check or change your current mode settings.
Pretty Print Mode

By default, **Pretty Print = ON**. Exponents, roots, fractions, etc., are displayed in the same form in which they are traditionally written. You can use \texttt{MODE} to turn pretty print off and on.

<table>
<thead>
<tr>
<th>Pretty Print</th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \pi^2, \frac{\pi}{2}, \sqrt{\frac{x-3}{2}} )</td>
<td>( \pi^2, \pi/2, \sqrt{(x-3)/2} )</td>
<td></td>
</tr>
</tbody>
</table>

The entry line does not show an expression in pretty print. If pretty print is turned on, the history area will show both the entry and its result in pretty print after you press \texttt{ENTER}.

Exact/Approx Mode

By default, **Exact/Approx = AUTO**. You can use \texttt{MODE} to select from three settings.

Because AUTO is a combination of the other two settings, you should be familiar with all three settings.
**EXACT** — Any result that is not a whole number is displayed in a fractional or symbolic form (1/2, π, \(\sqrt{2}\), etc.).

| 2 \cdot 2          | 5   |
| 2 \cdot 3          | 6/3 |
| 6/3                | 2   |
| 6/4                | 3/2 |

**Shows whole-number results.**

| \(\sqrt{2}\)         | 15/2 |
| \(\sqrt{4/7}\)       | 2/2  |
| \(\sqrt{4/7}\)       | 2/7  |

**Shows simplified fractional results.**

| 2 \cdot \pi           | 2 \cdot \pi |
| \sqrt{2}              | \sqrt{2} |
| \sqrt{4/7}            | 2 \cdot \sqrt{7} |

**Shows symbolic \(\pi\).**

| \sqrt{4/7}            | \sqrt{4/7} |
| \sqrt{4/7}            | .755929 |

**Shows symbolic form of roots that cannot be evaluated to a whole number.**

Press \(\bullet\) [ENTER] to temporarily override the EXACT setting and display a floating-point result.

**Note:** By retaining fractional and symbolic forms, EXACT reduces rounding errors that could be introduced by intermediate results in chained calculations.
**APPROXIMATE** — All numeric results, where possible, are displayed in floating-point (decimal) form.

**Note:** Results are rounded to the precision of your graphing calculator and displayed according to current mode settings.

Fractional results are evaluated numerically.

Symbolic forms, where possible, are evaluated numerically.

Because undefined variables cannot be evaluated, they are treated algebraically.
For example, if the variable \( r \) is undefined, \( \pi r^2 = 3.14159 \cdot r^2 \).

**AUTO** — Uses the EXACT form where possible, but uses the APPROXIMATE form when your entry contains a decimal point. Also, certain functions may display APPROXIMATE results even if your entry does not contain a decimal point.

A decimal in the entry forces a floating-point result.
**Note:** To retain an EXACT form, use fractions instead of decimals. For example, use 3/2 instead of 1.5.

The following chart compares the three settings.

<table>
<thead>
<tr>
<th>Entry</th>
<th>Exact Result</th>
<th>Approximate Result</th>
<th>Auto Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>8/4</td>
<td>2</td>
<td>2.</td>
<td>2</td>
</tr>
<tr>
<td>8/6</td>
<td>4/3</td>
<td>1.33333</td>
<td>4/3</td>
</tr>
<tr>
<td>8.5*3</td>
<td>51/2</td>
<td>25.5</td>
<td>25.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>√(2)/2</td>
<td>√2/2</td>
<td>.707107</td>
<td>√2/2</td>
</tr>
<tr>
<td>π*2</td>
<td>2·π</td>
<td>6.28319</td>
<td>2·π</td>
</tr>
<tr>
<td>π*2.</td>
<td>2·π</td>
<td>6.28319</td>
<td>6.28319</td>
</tr>
</tbody>
</table>

**Note:** To evaluate an entry in APPROXIMATE form, regardless of the current setting, press ENTER.

**Display Digits Mode**

By default, Display Digits = FLOAT 6, which means that results are rounded to a maximum of six digits. You can use MODE to select different settings. The settings apply to all exponential formats.
Internally, the calculator calculates and retains all decimal results with up to 14 significant digits (although a maximum of 12 are displayed).

<table>
<thead>
<tr>
<th>Setting</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIX 0–12</td>
<td>123.</td>
<td>(FIX 0) Results are rounded to the selected number of decimal places.</td>
</tr>
<tr>
<td></td>
<td>123.5</td>
<td>(FIX 1)</td>
</tr>
<tr>
<td></td>
<td>123.46</td>
<td>(FIX 2)</td>
</tr>
<tr>
<td></td>
<td>123.457</td>
<td>(FIX 3)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>123.456789012</td>
<td>Number of decimal places varies, depending on the result.</td>
</tr>
<tr>
<td>FLOAT 1–12</td>
<td>1.2E 2</td>
<td>(FLOAT 1) Results are rounded to the total number of selected digits.</td>
</tr>
<tr>
<td></td>
<td>123.</td>
<td>(FLOAT 2)</td>
</tr>
<tr>
<td></td>
<td>123.5</td>
<td>(FLOAT 3)</td>
</tr>
<tr>
<td></td>
<td>123.46</td>
<td>(FLOAT 4)</td>
</tr>
<tr>
<td></td>
<td>123.457</td>
<td>(FLOAT 5)</td>
</tr>
<tr>
<td></td>
<td>123.457</td>
<td>(FLOAT 6)</td>
</tr>
</tbody>
</table>

**Notes:**

- Regardless of the **Display Digits** setting, the full value is used for internal floating-point calculations to ensure maximum accuracy.
- A result is automatically shown in scientific notation if its magnitude cannot be displayed in the selected number of digits.

**Exponential Format Mode**

By default, **Exponential Format = NORMAL.** You can use **MODE** to select from three settings.
<table>
<thead>
<tr>
<th>Setting</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMAL</td>
<td>12345.6</td>
<td>If a result cannot be displayed in the number of digits specified by the Display Digits mode, the calculator switches from NORMAL to SCIENTIFIC for that result only.</td>
</tr>
<tr>
<td>SCIENTIFIC</td>
<td>1.23456E 4</td>
<td>1.23456 × 10^4</td>
</tr>
<tr>
<td>ENGINEERING</td>
<td>1.23456E 3</td>
<td>12.3456 × 10^3</td>
</tr>
</tbody>
</table>

1. Always 1 digit to the left of the decimal point.
2. Exponent (power of 10).
3. May have 1, 2, or 3 digits to the left of the decimal point.
4. Exponent is a multiple of 3.

**Note:** In the history area, a number in an entry is displayed in SCIENTIFIC if its absolute value is less than .001.

**Editing an Expression in the Entry Line**

Knowing how to edit an entry can be a real time-saver. If you make an error while typing an expression, it’s often easier to correct the mistake than to retype the entire expression.
Removing the Highlight from the Previous Entry

After you press [ENTER] to evaluate an expression, the calculator leaves that expression on the entry line and highlights it. To edit the expression, you must first remove the highlight; otherwise, you may clear the expression accidentally by typing over it.

To remove the highlight, move the cursor toward the side of the expression you want to edit.

 annunciator moves the cursor to the beginning.
 annunciator moves the cursor to the end of the expression.

Moving the Cursor

After removing the highlight, move the cursor to the applicable position within the expression.

<table>
<thead>
<tr>
<th>To move the cursor:</th>
<th>Press:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left or right within an expression.</td>
<td>annunciator or annunciator</td>
</tr>
<tr>
<td>Hold the pad to repeat the movement.</td>
<td></td>
</tr>
<tr>
<td>To the beginning of the expression.</td>
<td>2nd annunciator</td>
</tr>
<tr>
<td>To the end of the expression.</td>
<td>2nd annunciator</td>
</tr>
</tbody>
</table>

Note: If you accidentally press annunciator instead of annunciator or annunciator, the cursor moves up into the history area. Press [ESC] or press annunciator until the cursor returns to the entry line.
## Deleting a Character

<table>
<thead>
<tr>
<th>To delete:</th>
<th>Press:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The character to the left of the cursor.</td>
<td>←</td>
</tr>
<tr>
<td>The character to the right of the cursor.</td>
<td>⬇️ ←</td>
</tr>
<tr>
<td>All characters to the right of the cursor.</td>
<td>CLEAR (once only)</td>
</tr>
</tbody>
</table>

To delete: Press:

- The character to the left of the cursor. [←]
- The character to the right of the cursor. [↑ ←]
- All characters to the right of the cursor. [CLEAR (once only)]
- If there are no characters to the right of the cursor, [CLEAR] erases the entire entry line.

## Clearing the Entry Line

To clear the entry line, press:

- [CLEAR] if the cursor is at the beginning or end of the entry line.
- or –
- [CLEAR CLEAR] if the cursor is not at the beginning or end of the entry line. The first press deletes all characters to the right of the cursor, and the second clears the entry line.
**Inserting or Overtyping a Character**

The calculator has both an insert and an overtype mode. By default, the calculator is in the insert mode. To toggle between the insert and overtype modes, press **2nd [INS]**.

<table>
<thead>
<tr>
<th>If in:</th>
<th>The next character you type:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Insert mode</strong></td>
<td>Will be inserted at the cursor.</td>
</tr>
<tr>
<td>Thin cursor between characters</td>
<td></td>
</tr>
<tr>
<td><strong>OverType mode</strong></td>
<td>Will replace the highlighted character.</td>
</tr>
<tr>
<td>Cursor highlights a character</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Look at the cursor to see if you’re in insert or overtype mode.

**Replacing or Deleting Multiple Characters**

First, highlight the applicable characters. Then, replace or delete all the highlighted characters.
To highlight multiple characters:

1. Move the cursor to either side of the characters you want to highlight.

2. Hold $\uparrow$ and press $\circlearrowleft$ or $\circlearrowright$ to highlight characters left or right of the cursor.

To replace or delete the highlighted characters:

1. Type the new characters.
2. Press $\leftarrow$.

Note: When you highlight characters to replace, remember that some function keys automatically add an open parenthesis.

Menus

To leave the keyboard uncluttered, the calculator uses menus to access many operations. This section gives an overview of how to select an item from any menu. Specific menus
are described in the appropriate modules.

## Displaying a Menu

<table>
<thead>
<tr>
<th>Press:</th>
<th>To display:</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1, F2, etc.</td>
<td>A toolbar menu — Drops down from the toolbar at the top of most application screens. Lets you select operations useful for that application.</td>
</tr>
<tr>
<td>APPS</td>
<td><strong>Apps desktop</strong> or <strong>APPLICATIONS</strong> menu — Lets you select from a list of applications.</td>
</tr>
<tr>
<td>2nd [CHAR]</td>
<td><strong>CHAR</strong> menu — Lets you select from categories of special characters (Greek, math, etc.).</td>
</tr>
<tr>
<td>2nd [MATH]</td>
<td><strong>MATH</strong> menu — Lets you select from categories of math operations.</td>
</tr>
<tr>
<td>CATALOG</td>
<td><strong>CATALOG</strong> menu — Lets you select from a complete, alphabetic list of built-in functions and instructions. Also lets you select user-defined functions or Flash application functions (if any have been defined or loaded).</td>
</tr>
<tr>
<td>2nd [CUSTOM]</td>
<td><strong>CUSTOM</strong> menu — Lets you access a menu that you can customize to list any available function, instruction, or character. The calculator includes a default custom menu, which you can modify or redefine. Refer to the <em>Calculator Home Screen</em> and/or the <em>Programming</em> module for more information on the custom menu.</td>
</tr>
</tbody>
</table>
Selecting an Item from a Menu

To select an item from the displayed menu, either:

- Press the number or letter shown to the left of that item. For a letter on the TI-89 Titanium, press \[\text{alpha}\] and then a letter key.

- or -

- Use the cursor pad \(\triangleleft\) and \(\triangleright\) to highlight the item, and then press \[\text{ENTER}\]. (Note that pressing \(\triangleright\) from the first item moves the highlight to the last item, and vice versa.)

\[\text{F2-} \]

\[\begin{array}{|l|}
\hline
1: \text{solve} \\
2: \text{factor} \\
3: \text{expand} \\
4: \text{zeros} \\
5: \text{approx} \\
6: \text{conDenom} \\
7: \text{propFrac} \\
8: \text{lnSolve} \\
\hline
\end{array}\]

\(\downarrow\) indicates that a menu will drop down from the toolbar when you press \[\text{F2}\].

To select \textit{factor}, press 2 or \(\triangleleft\) \[\text{ENTER}\]. This closes the menu and inserts the function at the cursor location.

\[\text{factor}(\quad)\]

Items Ending with \(\triangleright\) (Submenus)

If you select a menu item ending with \(\triangleright\), a submenu is displayed. You then select an item from the submenu.
For items that have a submenu, you can use the cursor pad as described below.

- To display the submenu for the highlighted item, press \( \text{\textcircled{5}} \). (This is the same as selecting that item.)
- To cancel the submenu without making a selection, press \( \text{\textcircled{6}} \). (This is the same as pressing [ESC].)
- To wrap to the last menu item directly from the first menu item, press \( \text{\textcircled{3}} \). To wrap to the first menu item directly from the last menu item, press \( \text{\textcircled{4}} \).
Items Containing “...” (Dialog Boxes)

If you select a menu item containing “...” (ellipsis marks), a dialog box is displayed for you to enter additional information.

For example, **Save Copy As ...** displays a dialog box that prompts you to select a folder name and type a variable name.

→ indicates that you can press → to display and select from a menu.

An input box indicates that you must type a value. (Alpha-lock is automatically turned on for the TI-89 Titanium.)

After typing in an input box such as Variable, you must press **ENTER** twice to save the information and close the dialog box.

**Canceling a Menu**

To cancel the current menu without making a selection, press **ESC**. Depending on whether any submenus are displayed, you may need to press **ESC** several times to cancel all displayed menus.
Moving from One Toolbar Menu to Another

To move from one toolbar menu to another without making a selection, either:

- Press the key (F1, F2, etc.) for the other toolbar menu.
  – or –
- Use the cursor pad to move to the next (press ④) or previous (press ③) toolbar menu. Pressing ④ from the last menu moves to the first menu, and vice versa.

When using ④, be sure that an item with a submenu is not highlighted. If so, ④ displays that item’s submenu instead of moving to the next toolbar menu.

Example: Selecting a Menu Item

Round the value of π to three decimal places. Starting from a clear entry line on the Home screen:

1. Press 2nd [MATH] to display the MATH menu.

2. Press 1 to display the Number submenu. (Or press [ENTER] since the first item is automatically highlighted.)

3. Press 3 to select round. (Or press ◀ ◀ and [ENTER].)

4. Press 2nd [π] ③ ③ and then [ENTER] to evaluate the expression.

Selecting the function in Step 3 automatically typed round( on the entry line.
Selecting an Application

The graphing calculator has different applications that let you solve and explore a variety of problems. You can select an application from a menu, the Apps desktop, or you can access commonly used applications directly from the keyboard.

From the APPLICATIONS Menu

1. If the Apps desktop is off, press [APPS] to display a menu that lists the applications.  
   **Note:** To cancel the menu without making a selection, press [ESC].

2. Select an application. Either:
   - Use the cursor pad ⬇️ or ⬆️ to highlight the application and then press [ENTER].  
     – or –  
   - Press the number or letter for that application.

<table>
<thead>
<tr>
<th>Application</th>
<th>Lets you:</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlashApps</td>
<td>Display a list of Flash applications, if any.</td>
</tr>
<tr>
<td>Y= Editor</td>
<td>Define, edit, and select functions or equations for graphing.</td>
</tr>
<tr>
<td>Window Editor</td>
<td>Set window dimensions for viewing a graph.</td>
</tr>
<tr>
<td>Graph</td>
<td>Display graphs.</td>
</tr>
<tr>
<td>Application:</td>
<td>Lets you:</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Table</td>
<td>Display a table of variable values that correspond to an entered function.</td>
</tr>
<tr>
<td>Data/Matrix Editor</td>
<td>Enter and edit lists, data, and matrices. You can perform statistical calculations and graph statistical plots.</td>
</tr>
<tr>
<td>Program Editor</td>
<td>Enter and edit programs and functions.</td>
</tr>
<tr>
<td>Text Editor</td>
<td>Enter and edit a text session.</td>
</tr>
<tr>
<td>Numeric Solver</td>
<td>Enter an expression or equation, define values for all but one variable, and then solve for the unknown variable.</td>
</tr>
<tr>
<td>Home</td>
<td>Enter expressions and instructions, and perform calculations.</td>
</tr>
</tbody>
</table>

**From the Apps Desktop**

Press the first letter of the application name, or use the cursor keys to highlight an application icon on the Apps desktop and press `[ENTER]`. (If you press the first letter of the application and more than one application begins with that letter, the first one alphabetically is highlighted). The application either opens directly or displays a dialog box. (Your Apps desktop may vary from the one shown below.)
The most common dialog box lists these options for the application:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>Returns the screen displayed when you last viewed the App. (If there is no current file/variable for the selected App, this option defaults to New if you press [ENTER].)</td>
</tr>
<tr>
<td>Open</td>
<td>Lets you select an existing file.</td>
</tr>
<tr>
<td>New</td>
<td>Creates a new file with the name typed in the field.</td>
</tr>
</tbody>
</table>

Select an option and press [ENTER]. The application appears.

**Note:** The general term *variable* is used to refer to the application data files that you create.

Use any of these methods to return to the Apps desktop from within an application:

- Press [APPS].
- In full-screen mode, press [2nd] [QUIT].
- In split-screen mode, press [2nd] [QUIT] to open the full-screen view of the active application, then press [2nd] [QUIT] again.

To return to the last open application from the Apps desktop, press [2nd] [←].
From the Keyboard
You can access commonly used applications from the keyboard. On the TI-89 Titanium for example, $\boxed{\downarrow \boxed{\text{[Y=]}}}$ is the same as pressing $\boxed{\downarrow}$ and then $\boxed{\text{F1}}$. This guidebook uses the notation $\boxed{\downarrow \boxed{\text{[Y=]}}}$, similar to the notation used in second functions.

<table>
<thead>
<tr>
<th>Application</th>
<th>Press:</th>
</tr>
</thead>
</table>
| Home                 | $\boxed{\text{HOME}}$  
                     | $\boxed{\text{CALC HOME}}$  |
| Y= Editor            | $\boxed{\downarrow \boxed{\text{[Y=]}}}$         |
| Window Editor        | $\boxed{\downarrow \boxed{\text{[WINDOW]}}}$   |
| Graph                | $\boxed{\downarrow \boxed{\text{[GRAPH]}}}$     |
| Table Setup          | $\boxed{\downarrow \boxed{\text{[TBLSET]}}}$   |
| Table Screen         | $\boxed{\downarrow \boxed{\text{[TABLE]}}}$     |

Applications listed above $\text{[F1]}$, $\text{[F2]}$ etc., are printed in the same color as $\boxed{\downarrow}$.

**Setting Modes**

Modes control how numbers and graphs are displayed and interpreted. Mode settings are retained by the Constant Memory™ feature when the graphing calculator is turned off. All numbers, including elements of matrices and lists, are displayed according to the current mode settings.

**Checking Mode Settings**

Press $\boxed{\text{MODE}}$ to display the MODE dialog box, which lists the modes and their current settings.
There are three pages of mode listings. Press F1, F2, or F3 to quickly display a particular page.

Indicates you can scroll down to see additional modes.

> indicates that you can press © or ª to display and select from a menu.

**Note:** Modes that are not currently valid are dimmed. For example, on **Page 2, Split 2 App** is not valid when **Split Screen = FULL**. When you scroll through the list, the cursor skips dimmed settings.

### Changing Mode Settings

From the MODE dialog box:

1. Highlight the mode setting you want to change. Use © or ª (with F1, F2, or F3) to scroll through the list.

2. Press © or ª to display a menu that lists the valid settings. The current setting is highlighted.

3. Select the applicable setting. Either:
   - Use © or ª to highlight the setting and press ENTER.
• Press the number or letter for that setting.

**Note:** To cancel a menu and return to the **MODE** dialog box without making a selection, press **Esc**.

4. Change other mode settings, if necessary.

5. When you finish all your changes, press **Enter** to save the changes and exit the dialog box.

**Important:** If you press **Esc** instead of **Enter** to exit the **MODE** dialog box, any mode changes you made will be canceled.

---

**Overview of the Modes**

**Note:** For detailed information about a particular mode, look in the applicable section of this guidebook.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph</td>
<td>Type of graphs to plot: FUNCTION, PARAMETRIC, POLAR, SEQUENCE, 3D, or DE.</td>
</tr>
<tr>
<td>Current Folder</td>
<td>Folder used to store and recall variables. Unless you have created additional folders, only the MAIN folder is available. Refer to “Using Folders to Store Independent Sets of Variables” in <em>Calculator Home Screen</em>.</td>
</tr>
<tr>
<td>Display Digits</td>
<td>Maximum number of digits (FLOAT) or fixed number of decimal places (FIX) displayed in a floating-point result. Regardless of the setting, the total number of displayed digits in a floating-point result cannot exceed 12.</td>
</tr>
<tr>
<td>Angle</td>
<td>Units in which angle values are interpreted and displayed: RADIAN, DEGREE or GRADIAN.</td>
</tr>
<tr>
<td>Mode</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Exponential Format</td>
<td>Notation used to display results: NORMAL, SCIENTIFIC, or ENGINEERING.</td>
</tr>
<tr>
<td>Complex Format</td>
<td>Format used to display complex results, if any: REAL (complex results are not displayed unless you use a complex entry), RECTANGULAR, or POLAR.</td>
</tr>
<tr>
<td>Vector Format</td>
<td>Format used to display 2- and 3-element vectors: RECTANGULAR, CYLINDRICAL, or SPHERICAL.</td>
</tr>
<tr>
<td>Pretty Print</td>
<td>Turns the pretty print display feature OFF or ON.</td>
</tr>
<tr>
<td>Split Screen</td>
<td>Splits the screen into two parts and specifies how the parts are arranged: FULL (no split screen), TOP-BOTTOM, or LEFT-RIGHT. Refer to the <em>Split Screens</em> module.</td>
</tr>
<tr>
<td>Split 1 App</td>
<td>Application in the top or left side of a split screen. If you are not using a split screen, this is the current application.</td>
</tr>
<tr>
<td>Split 2 App</td>
<td>Application in the bottom or right side of a split screen. This is active only for a split screen.</td>
</tr>
<tr>
<td>Number of Graphs</td>
<td>For a split screen, lets you set up both sides of the screen to display independent sets of graphs.</td>
</tr>
<tr>
<td>Graph 2</td>
<td>If <strong>Number of Graphs</strong> = 2, selects the type of graph in the Split 2 part of the screen. Refer to <em>Calculator Home Screen</em>.</td>
</tr>
<tr>
<td>Exact/Approx</td>
<td>Calculates expressions and displays results in numeric form or in rational/symbolic form: AUTO, EXACT, or APPROXIMATE.</td>
</tr>
<tr>
<td>Base</td>
<td>Lets you perform calculations by entering numbers in decimal (DEC), hexadecimal (HEX), or binary (BIN) form.</td>
</tr>
</tbody>
</table>
### Clean Up Toolbar Menu

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit System</td>
<td>Lets you select from three systems of measurement to specify the default units for displayed results: <strong>SI</strong> (metric or MKS); <strong>Eng/US</strong> (feet, pounds, etc.); or <strong>Custom</strong>.</td>
</tr>
<tr>
<td>Custom Units</td>
<td>Lets you select custom defaults. The mode is dimmed until you select Unit System, <strong>3:CUSTOM</strong>.</td>
</tr>
<tr>
<td>Language</td>
<td>Lets you localize the calculator into one of several languages, depending on which language Flash applications are installed.</td>
</tr>
<tr>
<td>Apps Desktop</td>
<td>Turns the Apps desktop ON or OFF.</td>
</tr>
</tbody>
</table>

## Using the Clean Up Menu to Start a New Problem

On the Home screen, the **Clean Up** toolbar menu lets you start a new calculation from a cleared state without resetting the memory.

### Clean Up Toolbar Menu

To display the **Clean Up** menu from the Home screen, press:

\[2nd\] [F6]
<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear a–z</td>
<td>Clears (deletes) all single-character variable names in the current folder, unless the variables are locked or archived. You will be prompted to press [ENTER] to confirm the action.</td>
</tr>
</tbody>
</table>
|            | Single-character variable names are often used in symbolic calculations such as: \[
|            | \text{solve}(a\cdot x^2 + b\cdot x + c = 0, x) \]
|            | If any of the variables have already been assigned a value, your calculation may produce misleading results. To prevent this, you can select 1:Clear a–z before beginning the calculation. |

NewProb

| NewProb   | Places NewProb in the entry line. You must then press \[ENTER\] to execute the command. NewProb performs a variety of operations that let you begin a new problem from a cleared state without resetting the memory: |
|           | • Clears all single-character variable names in the current folder (same as 1:Clear a–z), unless the variables are locked or archived. |
|           | • Turns off all functions and stat plots (FnOff and PlotsOff) in the current graphing mode. |
|           | • Performs ClrDraw, ClrErr, ClrGraph, ClrHome, ClrIO, and ClrTable. |

Restore custom default

| Restore custom default | If a custom menu other than the default is in effect, this lets you restore the default. Refer to the Calculator Home Screen module for information on the custom menu. |

Notes:
• When defining a variable that you want to retain, use more than one character in the name. This prevents it from being deleted inadvertently by 1:Clear a–z.

• For information about checking and resetting memory or other system defaults, refer to Memory and Variable Management.

Using the Catalog Dialog Box

The CATALOG provides a way to access any built-in command (functions and instructions) from one convenient list. In addition, the CATALOG dialog box lets you select functions used in Flash applications or user-defined functions (if any have been loaded or defined).

Displaying the CATALOG

To display the CATALOG dialog box, press:

![CATALOG]

The CATALOG defaults to F2 Built-in, which displays an alphabetic list of all pre-installed commands (functions and instructions).
1. Defaults to F2 Built-in.
2. F1 Help displays a command’s parameters in a dialog box.
3. F3 and F4 allow access to Flash application functions and User-Defined functions and programs.

Note: Options that are not currently valid are dimmed. For example, F3 Flash Apps is dimmed if you have not installed a Flash application. F4 User-Defined is dimmed if you have not created a function or a program.

Selecting a Built-in Command from the CATALOG

When you select a command, its name is inserted in the entry line at the cursor location. Therefore, you should position the cursor as necessary before selecting the command.

1. Press:
   CATALOG

2. Press F2 Built-in.

- Commands are listed in alphabetical order. Commands that do not start with a letter (+, %, √, Σ, etc.) are at the end of the list.
- To exit the CATALOG without selecting a command, press [ESC].

Note: The first time you display the Built-in list, it starts at the top of the list. The next time you display the list, it starts at the same place you left it.
3. Move the ▶ indicator to the command, and press [ENTER].

<table>
<thead>
<tr>
<th>To move the ▶ indicator:</th>
<th>Press or type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>One function or program at a time</td>
<td>⇢ or ⇣</td>
</tr>
<tr>
<td>One page at a time</td>
<td>2nd ⇢ or 2nd ⇣</td>
</tr>
<tr>
<td>To the first function that begins with a specified letter</td>
<td>The letter key. (On the TI-89 Titanium, do not press [alpha] first. If you do, you need to press [alpha] or 2nd [a-lock] again before you can type a letter.)</td>
</tr>
</tbody>
</table>

**Note:** From the top of the list, press ⇢ to move to the bottom. From the bottom, press ⇣ to move to the top.

**Information about Parameters**

For the command indicated by ▶, the status line shows the required and optional parameters, if any, and their type.

1. Indicated command and its parameters
2. Brackets [ ] indicate optional parameters
From the example above, the syntax for factor is:

\[
\text{factor}(\text{expression}) \quad \text{required} \\
\text{or} \\
\text{factor}(\text{expression},\text{variable}) \quad \text{optional}
\]

**Note:** For details about the parameters, refer to that command’s description in the *Technical Reference* module.

**Viewing CATALOG Help**

You can display a command's parameters in a dialog box by pressing \( \text{F1 Help} \). The parameters are the same as those displayed on the status line.

Some commands, such as \text{ClrDraw}, do not require parameters. If you select one of these commands, parameters will not display on the status line and you will see Unavailable if you press \( \text{F1 Help} \).

Press \( \text{ESC} \) to exit the CATALOG Help dialog box.
Selecting a Flash Application Function

A Flash application may contain one or more functions. When you select a function, its name is inserted in the entry line at the cursor location. Therefore, you should position the cursor as necessary before selecting the function.

1. Press:
   
   CATALOG

2. Press F3 Flash Apps. (This option is dimmed if no Flash applications are installed.)

   - The list is alphabetized by function name. The left column lists functions. The right column lists the Flash application that contains the function.
   - Information about a function is displayed in the status line.
   - To exit without selecting a function, press ESC.

3. Move the ▶ indicator to the function, and press ENTER.

<table>
<thead>
<tr>
<th>To move the ▶ indicator:</th>
<th>Press or type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>One function or program at a time</td>
<td>↘ or ↗</td>
</tr>
<tr>
<td>One page at a time</td>
<td>2nd ↘ or 2nd ↗</td>
</tr>
<tr>
<td>To the first function that begins with a specified letter</td>
<td>The letter key. (On the TI-89 Titanium, do not press alpha first. If you do, you need to press alpha or 2nd [a-lock] again before you can type a letter.)</td>
</tr>
</tbody>
</table>
Selecting a User-Defined Function or Program

You can create your own functions or programs and then use [F4] User-Defined to access them. For instructions on how to create functions, see “Creating and Evaluating User-Defined Functions” in Calculator Home Screen, and “Overview of Entering a Function” in the Programming module. See Programming for instructions on how to create and run a program.

When you select a function or program, its name is inserted in the entry line at the cursor location. Therefore, you should position the cursor as necessary before selecting the function or program.

1. Press:
   CATALOG

2. Press [F4] User-Defined. (This option is dimmed if you have not defined a function or created a program.)

   - The list is alphabetized by function / program name. The left column lists functions and programs. The right column lists the folder that contains the function or program.
   - If the function or program's first line is a comment, the comment text is displayed in the status line.
   - To exit without selecting a function or program, press [ESC].

Note: Use the VAR-LINK screen to manage variables, folders, and Flash applications. See the Memory and Variable Management module.
3. Move the $\blacktriangleleft$ indicator to the function or program, and press $\text{[ENTER]}$.

<table>
<thead>
<tr>
<th>To move the $\blacktriangleleft$ indicator:</th>
<th>Press or type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>One function or program at a time</td>
<td>$\blacktriangleright$ or $\blacktriangleright$</td>
</tr>
<tr>
<td>One page at a time</td>
<td>$\text{2nd} \blacktriangleright$ or $\text{2nd} \blacktriangleright$</td>
</tr>
<tr>
<td>To the first function or program that begins with a specified letter</td>
<td>The letter key. (On the TI-89 Titanium, do not press $\text{[alpha]}$ first. If you do, you need to press $\text{[alpha]}$ or $\text{2nd} \text{[a-lock]}$ again before you can type a letter.)</td>
</tr>
</tbody>
</table>

**Storing and Recalling Variable Values**

When you store a value, you store it as a named variable. You can then use the name instead of the value in expressions. When the calculator encounters the name in an expression, it substitutes the variable’s stored value.

**Rules for Variable Names**

A variable name:

- Can use 1 to 8 characters consisting of letters and digits. This includes Greek letters (but not $\pi$), accented letters, and international letters.
  - Do not include spaces.
  - The first character cannot be a digit.
- Can use uppercase or lowercase letters. The names **AB22**, **Ab22**, **aB22**, and **ab22** all refer to the same variable.
- Cannot be the same as a name that is preassigned by the calculator. Preassigned names include:
  - Built-in functions (such as `abs`) and instructions (such as `LineVert`). Refer to the *Technical Reference* module.
  - System variables (such as `xmin` and `xmax`, which are used to store graph-related values). Refer to the *Technical Reference* module for a list.

**Examples**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>myvar</td>
<td>OK</td>
</tr>
<tr>
<td>a</td>
<td>OK</td>
</tr>
<tr>
<td>Log</td>
<td>Not OK, name is preassigned to the <code>log</code> function.</td>
</tr>
<tr>
<td>Log1</td>
<td>OK</td>
</tr>
<tr>
<td>3rdTotal</td>
<td>Not OK, starts with a digit.</td>
</tr>
<tr>
<td>circumfer</td>
<td>Not OK, more than 8 characters.</td>
</tr>
</tbody>
</table>

**Data Types**

<table>
<thead>
<tr>
<th>DataTypes</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expressions</td>
<td>2.54, 1.25E6, 2π, xmin/10, 2+3i, (x−2)^2, (\sqrt{2}/2)</td>
</tr>
<tr>
<td>Lists</td>
<td>{2 4 6 8}, {1 1 2}</td>
</tr>
</tbody>
</table>
Storing a Value in a Variable

1. Enter the value you want to store, which can be an expression.
2. Press \(\text{STO} \rightarrow\). The store symbol (\(\rightarrow\)) is displayed.
3. Type the variable name.

   **Note:** TI-89 Titanium users should use \(\alpha\) as necessary when typing variable names.

4. Press \(\text{ENTER}\).

To store to a variable temporarily, you can use the “with” operator. Refer to “Substituting Values and Setting Constraints” in *Symbolic Manipulation*.
Displaying a Variable

1. Type the variable name.
2. Press \[ \text{ENTER} \].

If the variable is undefined, the variable name is shown in the result.

In this example, the variable \( a \) is undefined. Therefore, it is used as a symbolic variable.

Note: Refer to Symbolic Manipulation for information about symbolic manipulation.

Using a Variable in an Expression

1. Type the variable name into the expression.
2. Press \[ \text{ENTER} \] to evaluate the expression.
   
   Note: To view a list of existing variable names, use \[ 2\text{nd} \ [ \text{VAR-LINK} \] as described in Memory and Variable Management.
   
   The variable’s value did not change.

If you want the result to replace the variable’s previous value, you must store the result.
Recalling a Variable's Value

In some cases, you may want to use a variable’s actual value in an expression instead of the variable name.

1. Press 2nd [RCL] to display a dialog box.
2. Type the variable name.
3. Press ENTER twice.

In this example, the value stored in num1 will be inserted at the cursor position in the entry line.

Status Line Indicators in the Display

The status line is displayed at the bottom of all application screens. It shows information about the current state of the calculator, including several important mode settings.

Status Line Indicators

1. Current Folder
2. Modifier Key
3. Angle Mode
### Indicator | Meaning
--- | ---
Current Folder | Shows the name of the current folder. Refer to “Using Folders to Store Independent Sets of Variables” in Calculator Home Screen. MAIN is the default folder.
Modifier Key | Shows which modifier key is in effect, as described below.

<table>
<thead>
<tr>
<th>Modifier Key</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd</td>
<td>will use the second function of the next key you press.</td>
</tr>
<tr>
<td>♦</td>
<td>will use the diamond feature of the next key you press.</td>
</tr>
<tr>
<td>↑</td>
<td>will type the uppercase letter for the next key you press. On the TI-89 Titanium, you can use ↑ to type a letter without having to use alpha.</td>
</tr>
<tr>
<td>↓</td>
<td>will type the lowercase letter for the next key you press.</td>
</tr>
<tr>
<td>[2nd a-lock]</td>
<td>lowercase alpha-lock is on. Until you turn this off, will type the lowercase letter for each key you press. To cancel alpha-lock, press alpha.</td>
</tr>
<tr>
<td>[↑ alpha]</td>
<td>uppercase ALPHA-lock is on. Until you turn this off, will type the uppercase letter for each key you press. To cancel ALPHA-lock, press alpha.</td>
</tr>
<tr>
<td>Indicator</td>
<td>Meaning</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Angle Mode</td>
<td>Shows the units in which angle values are interpreted and displayed. To change the Angle mode, use the [MODE] key.</td>
</tr>
<tr>
<td>RAD</td>
<td>Radians</td>
</tr>
<tr>
<td>DEG</td>
<td>Degrees</td>
</tr>
<tr>
<td>GRAD</td>
<td>Gradian</td>
</tr>
<tr>
<td>Exact/Approx Mode</td>
<td>Shows how answers are calculated and displayed. To change the Exact/Approx mode, use the [MODE] key.</td>
</tr>
<tr>
<td>AUTO</td>
<td>Auto</td>
</tr>
<tr>
<td>EXACT</td>
<td>Exact</td>
</tr>
<tr>
<td>APPROX</td>
<td>Approximate</td>
</tr>
<tr>
<td>Graph Number</td>
<td>If the screen is split to show two independent graphs, this indicates which graph is active — G1 or G2.</td>
</tr>
<tr>
<td>Graph Mode</td>
<td>Indicates the type of graphs that can be plotted. To change the Graph mode, use the [MODE] key.</td>
</tr>
<tr>
<td>FUNC</td>
<td>y(x) functions</td>
</tr>
<tr>
<td>PAR</td>
<td>x(t) and y(t) parametric equations</td>
</tr>
<tr>
<td>POL</td>
<td>r(θ) polar equations</td>
</tr>
<tr>
<td>SEQ</td>
<td>u(n) sequences</td>
</tr>
<tr>
<td>3D</td>
<td>z(x,y) 3D equations</td>
</tr>
<tr>
<td>DE</td>
<td>y'(t) differential equations</td>
</tr>
<tr>
<td>Indicator</td>
<td>Meaning</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Battery</td>
<td>Displayed only when the batteries are getting low. If BATT is shown with a black background, change the batteries as soon as possible.</td>
</tr>
<tr>
<td>History Pairs, Busy/Pause, Archived</td>
<td>The information shown in this part of the status line depends on the application you are using.</td>
</tr>
<tr>
<td>23/30</td>
<td>Displayed on the Home screen to show the number of entry/answer pairs in the history area. Refer to History Information on the Status Line in the <em>Calculator Home Screen</em> module.</td>
</tr>
<tr>
<td>BUSY</td>
<td>A calculation or graph is in progress.</td>
</tr>
<tr>
<td>PAUSE</td>
<td>You paused a graph or program.</td>
</tr>
<tr>
<td>📜</td>
<td>The variable opened in the current editor (Data/Matrix Editor, Program Editor, or Text Editor) is locked or archived and cannot be modified.</td>
</tr>
</tbody>
</table>

**Notes:**

- To cancel `2nd`, `♦`, `alpha`, or `↑`, press the same key again or press a different modifier key.
- If the next key you press does not have a diamond feature or an associated letter, the key performs its normal operation.
The calculator Home screen is the starting point for math operations, including executing instructions, evaluating expressions, and viewing results.

A blank calculator Home screen

This module describes the parts of the calculator Home screen, how to scroll through or modify the history area; how to use cut, copy, and paste, and more.

**Note:** The term “calculator Home screen” is used in this module. Other modules use the term “Home screen.” Both terms are interchangeable and refer to the same screen.

**Displaying the Calculator Home Screen**

When you first turn on your TI-89 Titanium, the Apps desktop is displayed. To display the calculator Home screen, highlight the Home icon and press [ENTER]. You can also display the calculator Home screen by pressing [HOME] (TI-89 Titanium). If you turn off the Apps Desktop mode, the calculator Home screen appears automatically.
Parts of the Calculator Home Screen

The following example contains previously entered data and describes the main parts of the calculator Home screen. Entry/answer pairs in the history area are displayed in “pretty print.” Pretty print displays expressions in the same form in which they are written on the board or in textbooks.

1 Toolbar
Let's you display menus for selecting operations applicable to the calculator Home screen. To display a toolbar menu, press \([F1], [F2], \) etc.

2 Pretty Print Display
Shows exponents, roots, fractions, etc., in traditional form.

3 Last Entry
Your last entry.

4 Entry Line
Where you enter expressions or instructions.

5 Status Line
Shows the current state of the calculator, including several important mode settings.
6 Last Answer
Result of your last entry. Note that results are not displayed on the entry line. Note: ENTER (Approx) was used in this example.

The following example shows an answer that is not on the same line as the expression. Note that the answer is longer than the screen width. An arrow (►) indicates the answer is continued. The entry line contains ellipsis (...). Ellipsis indicates the entry is longer than the screen width.

1 Last Entry
"Pretty print" is ON. Exponents, roots, fractions, etc., are displayed in the same form in which they are traditionally written.

2 History Area
Lists entry/answer pairs you have entered. Pairs scroll up the screen as you make new entries.

3 Answer Continues
Highlight the answer and press ◄ to scroll right and view the rest of it. Note that the answer is not on the same line as the expression.

4 Expression Continues (...)
Press ◄ to scroll right and view the rest of the entry. Press 2nd ◄ or 2nd ◄ to go to the beginning or end of the entry line.
### History Area

The history area shows up to eight previous entry/answer pairs (depending on the complexity and height of the displayed expressions). When the display is filled, information scrolls off the top of the screen. You can use the history area to:

- Review previous entries and answers. You can use the cursor to view entries and answers that have scrolled off the screen.
- Recall or auto-paste a previous entry or answer onto the entry line so that you can re-use or edit it.

### Scrolling through the History Area

Normally, the cursor is in the entry line. However, you can move the cursor into the history area.

<table>
<thead>
<tr>
<th>To:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>View entries or answers that have scrolled off the screen</td>
<td>- From the entry line, press ✈ to highlight the last answer.&lt;br&gt; - Continue using ✈ to move the cursor from answer to entry, up through the history area.</td>
</tr>
</tbody>
</table>

<p>| Go to the oldest or newest history pair | If the cursor is in the history area, press ✈ or ✐, respectively. |
| View an entry or answer that is too long for one line (▶ is at end of line) | Move the cursor to the entry or answer. Use ‣ and ‡ to scroll left and right (or 2nd ‣ and 2nd ‡ to go to the beginning or end), respectively. |</p>
<table>
<thead>
<tr>
<th>To:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return the cursor to the entry line</td>
<td>Press [ESC], or press ☐ until the cursor is back on the entry line.</td>
</tr>
</tbody>
</table>

**Note:** An example of viewing a long answer is available.

**History Information on the Status Line**

Use the history indicator on the status line for information about the entry/answer pairs. For example:

- **If the cursor is on the entry line:**
  - Total number of pairs that are currently saved.
  - Maximum number of pairs that can be saved.

- **If the cursor is in the history area:**
  - Pair number of the highlighted entry or answer.
  - Total number of pairs that are currently saved.

By default, the last 30 entry/answer pairs are saved. If the history area is full when you make a new entry (indicated by 30/30), the new entry/answer pair is saved and the oldest pair is deleted. The history indicator does not change.
Modifying the History Area

<table>
<thead>
<tr>
<th>To:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change the number of pairs that can be saved</td>
<td>Press F1 and select 9:Format, or press [ \leftrightarrow ] [ \downarrow ] [ \leftarrow ]</td>
</tr>
<tr>
<td></td>
<td>Then press [ \uparrow ], use [ \leftrightarrow ] or [ \leftarrow ] to highlight the new number, and press ENTER twice.</td>
</tr>
<tr>
<td>Clear the history area and delete all saved pairs</td>
<td>Press F1 and select 8:Clear Home, or enter ClrHome on the entry line.</td>
</tr>
<tr>
<td>Delete a particular entry/answer pair</td>
<td>Move the cursor to either the entry or answer. Press [ \leftarrow ] or [ CLEAR ].</td>
</tr>
</tbody>
</table>

Saving the Calculator Home Screen Entries as a Text Editor Script

To save all the entries in the history area, you can save the calculator Home screen to a text variable. When you want to reexecute those entries, use the Text Editor to open the variable as a command script.

Saving the Entries in the History Area

From the calculator Home screen:

1. Press F1 and select 2:Save Copy As.
2. Specify a folder and text variable that you want to use to store the entries.

**Note:** Only the entries are saved, not the answers.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Automatically set as Text and cannot be changed.</td>
</tr>
<tr>
<td>Folder</td>
<td>Shows the folder in which the text variable will be stored. To use a different folder, press ( ) to display a menu of existing folders. Then select a folder.</td>
</tr>
<tr>
<td>Variable</td>
<td>Type a valid, unused variable name.</td>
</tr>
</tbody>
</table>

**Note:** For information about folders, see the *Memory and Variable Management module*.

3. Press **ENTER** (after typing in an input box such as `Variable`, press **ENTER** twice).

**Restoring the Saved Entries**

Because the entries are stored in a script format, you cannot restore them from the calculator Home screen. (On the calculator Home screen’s \( \text{F1} \) toolbar menu, **1:Open** is not available.) Instead:
1. Use the Text Editor to open the variable containing the saved calculator Home screen entries.

The saved entries are listed as a series of command lines that you can execute individually, in any order.

2. Starting with the cursor on the first line of the script, press F4 repeatedly to execute the commands line by line.

3. Display the restored calculator Home screen.

This split screen shows the Text Editor (with the command line script) and the restored calculator Home screen.

Note: For complete information on using the Text Editor and executing a command script, refer to the Text Editor module.

**Cutting, Copying, and Pasting Information**

Cut, copy, and paste operations let you move or copy information within the same application or between different applications. These operations use the clipboard, which is an area in memory that serves as a temporary storage location.

**Auto-paste vs. Cut/Copy/Paste**

Auto-paste is a quick way to copy an entry or answer in the history area and paste it to the entry line.
1. Use ⬆️ and ⬇️ to highlight the item in the history area.
2. Press ENTER to auto-paste that item to the entry line.

To copy or move information in the entry line, you must use a cut, copy, or paste operation. (You can perform a copy operation in the history area, but not a cut or paste.)

**Cutting or Copying Information to the Clipboard**

When you cut or copy information, that information is placed in the clipboard. However, cutting deletes the information from its current location (used to move information) and copying leaves the information.

1. Highlight the characters that you want to cut or copy.
   
   In the entry line, move the cursor to either side of the characters. Hold ↑️ and press ◀️ or ▶️ to highlight characters to the left or right of the cursor, respectively.

2. Press F1 and select 4:Cut or 5:Copy.
Note: You can cut, copy or paste without having to use the [F1] toolbar menu. Press: [OS][CUT], [OS][COPY], or [OS][PASTE]

Cutting is not the same as deleting. When you delete information, it is not placed in the clipboard and cannot be retrieved.

Note: When you cut or copy information, it replaces the clipboard’s previous contents, if any.

Pasting Information from the Clipboard

A paste operation inserts the contents of the clipboard at the current cursor location on the entry line. This does not change the contents of the clipboard.

1. Position the cursor where you want to paste the information.
2. Press \[ F1 \] and select 6:Paste, or use the key shortcut:
\[ \text{복사} \quad \text{복사} \ [\text{PASTE}] \]

### Example: Copying and Pasting

Suppose you want to reuse an expression without retyping it each time.

1. Copy the applicable information.
   
   a) Use \[ \text{상하로 이동} \] or \[ \text{좌우로 이동} \] to highlight the expression.
   
   b) Press:
   \[ \text{복사} \quad \text{복사} \ [\text{COPY}] \]
   
   c) For this example, press \[ \text{ENTER} \] to evaluate the entry.

2. Paste the copied information into a new entry.
   
   a) Begin a new entry and place the cursor where you want to paste the copied information.
b) Press \( \text{F3 } 1 \) to select the \( d \) (differentiate) function.

c) Press:

\[ \begin{array}{c}
\text{PASTE} \\
\hline
\end{array} \]

to paste the copied expression.

d) Complete the new entry, and press \( \text{ENTER} \).

Note: You can also reuse an expression by creating a user-defined function.

3. Paste the copied information into a different application.

a) Press \( \text{Y=} \) to display the \( Y= \) Editor.

b) Press \( \text{ENTER} \) to define \( y_1(x) \).

c) Press:

\[ \begin{array}{c}
\text{PASTE} \\
\hline
\end{array} \]

to paste.

d) Press \( \text{ENTER} \) to save the new definition.

Note: By copying and pasting, you can easily transfer information from one application to another.
Reusing a Previous Entry or the Last Answer

You can reuse a previous entry by reexecuting the entry “as is” or by editing the entry and then reexecuting it. You can also reuse the last calculated answer by inserting it into a new expression.

Reusing the Expression on the Entry Line

When you press \[\text{ENTER}\] to evaluate an expression, the TI-89 Titanium leaves that expression on the entry line and highlights it. You can type over the entry, or you can reuse it as necessary.

For example, using a variable, find the square of 1, 2, 3, etc. As shown below, set the initial variable value and then enter the variable expression. Next, reenter to increment the variable and calculate the square.

<table>
<thead>
<tr>
<th>TI-89 Titanium</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 [\text{STO}]</td>
<td>(0 \div \text{num}) 0</td>
</tr>
<tr>
<td>2nd [a-lock] NUM</td>
<td>(0 \div \text{num}) 0</td>
</tr>
<tr>
<td>\text{ENTER}</td>
<td>(0 \div \text{num}) 0</td>
</tr>
<tr>
<td>NUM alpha + 1 STO</td>
<td>(0 \div \text{num}) 0</td>
</tr>
<tr>
<td>2nd [a-lock] NUM</td>
<td>(0 \div \text{num}) 0</td>
</tr>
<tr>
<td>2nd [:] NUM 2</td>
<td>(0 \div \text{num}) 0</td>
</tr>
<tr>
<td>\text{ENTER}</td>
<td>(0 \div \text{num}) 0</td>
</tr>
</tbody>
</table>
**Note:** Reexecuting an entry “as is” is useful for iterative calculations that involve variables.

Using the equation $A = \pi r^2$, use trial and error to find the radius of a circle that covers 200 square centimeters.

**Note:** Editing an entry lets you make minor changes without retyping the entire entry.

The example below uses 8 as the first guess and then displays the answer in its approximate floating-point form. You can edit and reexecute using 7.95 and continue until the answer is as accurate as you want.

<table>
<thead>
<tr>
<th>TI-89 Titanium</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 [STO⇒ alpha R 2nd [ : ] 2nd [π] alpha R ^ 2 ENTER</td>
<td><img src="image" alt="Display" /></td>
</tr>
</tbody>
</table>

```plaintext
8 \pi \times (r)^2 = 201.062
```
Note: When the entry contains a decimal point, the result is automatically displayed in floating-point.

Recalling a Previous Entry

You can recall any previous entry that is stored in the history area, even if the entry has scrolled off the top of the screen. The recalled entry replaces whatever is currently shown on the entry line. You can then reexecute or edit the recalled entry.

<table>
<thead>
<tr>
<th>To recall:</th>
<th>Press:</th>
<th>Effect:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The last entry (if you’ve changed the entry line)</td>
<td>2nd ENTRY] once</td>
<td>If the last entry is still shown on the entry line, this recalls the entry prior to that.</td>
</tr>
<tr>
<td>Previous entries</td>
<td>2nd ENTRY] repeatedly</td>
<td>Each press recalls the entry prior to the one shown on the entry line.</td>
</tr>
</tbody>
</table>

Note: You can also use the entry function to recall any previous entry. Refer to entry( ) in the Technical Reference module.

For example:
Recalling the Last Answer

Each time you evaluate an expression, the TI-89 Titanium stores the answer to the variable `ans(1)`. To insert this variable in the entry line, press `2nd` [ANS].

For example, calculate the area of a garden plot that is 1.7 meters by 4.2 meters. Then calculate the yield per square meter if the plot produces a total of 147 tomatoes.

1. Find the area.
   
   \[1.7 \times 4.2 \text{ ENTER}\]

2. Find the yield.
   
   \[147 \div 2 \text{nd} \text{ [ANS]} \text{ ENTER}\]

Just as `ans(1)` always contains the last answer, `ans(2), ans(3),` etc., also contain previous answers. For example, `ans(2)` contains the next-to-last answer.

**Note:** Refer to `ans()` in the Technical Reference module.
Auto-Pasting an Entry or Answer from the History Area

You can select any entry or answer from the history area and “auto-paste” a duplicate of it on the entry line. This lets you insert a previous entry or answer into a new expression without having to retype the previous information.

Why Use Auto-Paste

The effect of using auto-paste is similar to 2nd [ENTRY] and 2nd [ANS] as described in the previous section, but there are differences.

<table>
<thead>
<tr>
<th>For entries:</th>
<th>Pasting lets you:</th>
<th>2nd [ENTRY] lets you:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Insert any previous entry into the entry line.</td>
<td>Replace the contents of the entry line with any previous entry.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For answers:</th>
<th>Pasting lets you:</th>
<th>2nd [ANS] lets you:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Insert the displayed value of any previous answer into the entry line.</td>
<td>Insert the variable ans(1), which contains the last answer only. Each time you enter a calculation, ans(1) is updated to the latest answer.</td>
</tr>
</tbody>
</table>

Note: You can also paste information by using the F1 toolbar menu.
Auto-Pasting an Entry or Answer

1. On the entry line, place the cursor where you want to insert the entry or answer.
2. Press \( \uparrow \) to move the cursor up into the history area. This highlights the last answer.
3. Use \( \uparrow \) and \( \downarrow \) to highlight the entry or answer to auto-paste.

• \( \uparrow \) moves from answer to entry up through the history area.
• You can use \( \uparrow \) to highlight items that have scrolled off the screen

**Note:** To cancel auto-paste and return to the entry line, press [ESC]. To view an entry or answer too long for one line (indicated by \( \uparrow \) at the end of the line), use \( \uparrow \) and \( \downarrow \) or [2nd] \( \uparrow \) and [2nd] \( \downarrow \).

4. Press [ENTER].

The highlighted item is inserted in the entry line.

This pastes the entire entry or answer. If you need only a part of the entry or answer, edit the entry line to delete the unwanted parts.
Creating and Evaluating User-Defined Functions

User-defined functions can be a great time-saver when you need to repeat the same expression (but with different values) multiple times. User-defined functions can also extend your TI-89 Titanium’s capabilities beyond the built-in functions.

Format of a Function

The following examples show user-defined functions with one argument and two arguments. You can use as many arguments as necessary. In these examples, the definition consists of a single expression (or statement).

\[
\begin{align*}
\text{cube}(x) &= x^3 \\
\text{xroot}(x,y) &= y^{1/x}
\end{align*}
\]

1. Function name
2. Argument list
3. Definition

When defining functions and programs, use unique names for arguments that will not be used in the arguments for a subsequent function or program call.

Note: Function names follow the same rules as variable names. Refer to “Storing and Recalling Variable Values” in Operating the Calculator.

In the argument list, be sure to use the same arguments that are used in the definition. For example, \textit{cube}(n) = x^3 gives unexpected results when you evaluate the function.
Arguments (x and y in these examples) are placeholders that represent whatever values you pass to the function. They do not represent the variables x and y unless you specifically pass x and y as the arguments when you evaluate the function.

---

**Creating a User-Defined Function**

Use one of the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>STO&gt;</code></td>
<td>Store an expression to a function name (including the argument list).</td>
</tr>
<tr>
<td><code>Define</code> command</td>
<td>Define a function name (including the argument list) as an expression.</td>
</tr>
<tr>
<td><code>Program Editor</code></td>
<td>Refer to <em>Programming</em> or information on creating a user-defined function.</td>
</tr>
</tbody>
</table>

---

```plaintext
Define `cube(x) = x^3`    Done
1
Define `xroot(x,y) = y^x`  Done

Define `xroot(x,y) = y^(1/x)`  Done
```

---
Creating a Multi-Statement Function

You can also create a user-defined function whose definition consists of multiple statements. The definition can include many of the control and decision-making structures (If, ElseIf, Return, etc.) used in programming.

Note: For information about similarities and differences between functions and programs, refer to Programming.

For example, suppose you want to create a function that sums a series of reciprocals based on an entered integer (n):

\[
\frac{1}{n} + \frac{1}{n-1} + \ldots + \frac{1}{1}
\]

When creating the definition of a multi-statement function, it may be helpful to visualize it first in a block form.

```
1 Func
2 Local temp, i
   If fPart(nn) ≠ 0 or nn ≤ 0
3     Return “bad argument”
4     0->temp
5 For i, nn, 1, -1
     approx(temp+1/i) -> temp
   EndFor
6 Return temp
7 EndFunc
```

1 Func and EndFunc must begin and end the function.
2 Variables not in the argument list must be declared as local.
3 Returns a message if \( nn \) is not an integer or if \( nn \leq 0 \).
4 Sums the reciprocals.
5 Returns the sum.

When entering a multi-statement function on the calculator Home screen, you must enter the entire function on a single line. Use the **Define** command just as you would for a single-statement function.

Use a colon to separate each statement.

```
Define sumrecip(nn)=Func:Local temp,i: ... :EndFunc
```

Use argument names that will never be used when calling the function or program.

On the calculator Home screen:

```
Define sumrecip(nn)=Func
```

Multi-statement functions show as

```
Define sumrecip(nn)=Func
```

Enter a multi-statement function on one line. Be sure to include colons.

**Note:** It’s easier to create a complicated multi-statement function in the Program Editor than on the calculator Home screen. Refer to *Programming*. 
### Evaluating a Function

You can use a user-defined function just as you would any other function. Evaluate it by itself or include it in another expression.

### Displaying and Editing a Function Definition

<table>
<thead>
<tr>
<th>To:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display a list of all user-defined functions</td>
<td>Press 2nd [VAR-LINK] to display the VAR-LINK screen. You may need to use the F2 View toolbar menu to specify the Function variable type. (Refer to Memory and Variable Management.) – or – Press: [CATALOG] F4</td>
</tr>
<tr>
<td>Display a list of Flash application functions</td>
<td>Press: [CATALOG] F3</td>
</tr>
</tbody>
</table>
### Display the definition of a user-defined function

<table>
<thead>
<tr>
<th>To:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>From the VAR-LINK screen, highlight the function and display the Contents menu.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2nd [F6]</td>
</tr>
<tr>
<td></td>
<td>– or –</td>
</tr>
<tr>
<td>From the calculator Home screen, press 2nd [RCL]. Type the function name but not the argument list (such as xroot), and press ENTER twice.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– or –</td>
</tr>
<tr>
<td>From the Program Editor, open the function. (Refer to Programming.)</td>
<td></td>
</tr>
</tbody>
</table>

### Edit the definition

<table>
<thead>
<tr>
<th>To:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>From the calculator Home screen, use 2nd [RCL] to display the definition. Edit the definition as necessary. Then use [STO] or Define to save the new definition.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– or –</td>
</tr>
<tr>
<td>From the Program Editor, open the function, edit it, and save your changes. (Refer to Programming.)</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** You can view a user-defined function in the CATALOG dialog box, but you cannot use the CATALOG to view or edit its definition.

### If an Entry or Answer Is “Too Big”

In some cases, an entry or answer may be “too long” and/or “too tall” to be displayed completely in the history area. In other cases, the TI-89 Titanium may not be able to display an answer because there is not enough free memory.
If an Entry or Answer Is “Too Long”

Move the cursor into the history area, and highlight the entry or answer. Then use the cursor pad to scroll. For example:

- The following shows an answer that is too long for one line.

  Press 1 or 2nd 1 to scroll left.  
  Press 0 or 2nd 0 to scroll right.  

- The following shows an answer that is both too long and too tall to be displayed on the screen.

  **Note:** This example uses the *randMat* function to generate a 25 x 25 matrix.

  1 Press 1 or 1 1 to scroll up
  2 Press 1 1 to scroll down
  3 Press 1 or 2nd 1 to scroll left
  4 Press 0 or 2nd 0 to scroll right

If There Is not Enough Memory

A << ...>> symbol is displayed when the TI-89 Titanium does not have enough free memory to display the answer.
For example:

```
seq(n, n, 1, 2500)  <<...
seq(n, n, 1, 2500)
MAIN  RUN AUTO  PUNG  1/30
```

**Note:** This example uses the seq function to generate a sequential list of integers from 1 to 2500.

When you see the `<< ...>>` symbol, the answer cannot be displayed even if you highlight it and try to scroll.

In general, you can try to:

- Free up additional memory by deleting unneeded variables and/or Flash applications. Use [2nd] [VAR-LINK] as described in *Memory and Variable Management*.
- If possible, break the problem into smaller parts that can be calculated and displayed with less memory.

**Using the Custom Menu**

The TI-89 Titanium has a custom menu that you can turn on and off at any time. You can use the default custom menu or create your own as described in the *Programming* module.
Turning the Custom Menu On and Off

When you turn on the custom menu, it replaces the normal toolbar menu. When you turn it off, the normal menu returns. For example, from the calculator Home screen’s normal toolbar menu, press \text{2nd} \ [\text{CATALOG}] to toggle the custom menu on and off.

Note: You can also turn the custom menu on and off by entering \text{CustmOn} or \text{CustmOff} in the entry line and pressing \text{ENTER}.

Unless the menu has been modified, the default custom menu appears.

<table>
<thead>
<tr>
<th>Menu</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1 Var</td>
<td>Common variable names.</td>
</tr>
<tr>
<td>F2 f(x)</td>
<td>Function names such as f(x), g(x), and f(x,y).</td>
</tr>
<tr>
<td>F3 Solve</td>
<td>Items related to solving equations.</td>
</tr>
<tr>
<td>F4 Unit</td>
<td>Common units such as _m, _ft, and _l.</td>
</tr>
<tr>
<td>F5 Symbol</td>
<td>Symbols such as #, ?, and ~.</td>
</tr>
</tbody>
</table>
**Note:** A custom menu can give you quick access to commonly used items. The *Programming* module shows you how to create custom menus for the items you use most often.

### Restoring the Default Custom Menu

If a custom menu other than the default is displayed and you want to restore the default:

1. From the calculator Home screen, use \(2\text{nd} \ [\text{CATALOG}]\) to turn off the custom menu and display the calculator Home screen’s normal toolbar menu.

2. Display the **Clean Up** toolbar menu, and select **3:Restore custom default**.  

```
<table>
<thead>
<tr>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
<th>F6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu Function</td>
<td>Menu Function</td>
<td>Menu Function</td>
<td>Menu Function</td>
<td>Menu Function</td>
<td>Menu Function</td>
</tr>
</tbody>
</table>
```

This pastes the commands used to create the default menu into the entry line.

**Note:** The previous custom menu is erased. If that menu was created with a program, it can be recreated later by running the program again.

3. Press \(\text{ENTER}\) to execute the commands and restore the default.
Finding the Software Version and ID Number

In some situations, you may need to find out information about your TI-89 Titanium, particularly the software version and the unit’s ID number.

Displaying the “About” Screen

1. From either the calculator Home screen or the Apps desktop, press \text{F1} and then select A:About.

   Your screen will be different from the one shown to the right.

2. Press \text{ENTER} or \text{ESC} to close the screen.

When Do You Need this Information?

The information on the About screen is intended for situations such as:

- If you obtain new or upgraded software or Flash applications for your TI-89 Titanium, you may need to provide your current software version and/or the ID number of your unit.
- If you have difficulties with your TI-89 Titanium and need to contact technical support, knowing the software version may make it easier to diagnose the problem.
The About screen displays the following information about your calculator:

- Hardware version
- OS (Advanced Mathematics Software) version
- Product identifier (Product ID)
- Unit ID
- Apps certificate revision number (Cert. Rev.)

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OS version</td>
</tr>
<tr>
<td>2</td>
<td>Product identifier</td>
</tr>
<tr>
<td>3</td>
<td>Apps certificate revision number</td>
</tr>
<tr>
<td>4</td>
<td>Hardware version</td>
</tr>
<tr>
<td>5</td>
<td>Unit ID (required to obtain certificates for installing purchased Apps)</td>
</tr>
</tbody>
</table>

Your screen will be different from the one shown above.
Symbolic Manipulation

Using Undefined or Defined Variables

When performing algebraic or calculus operations, it is important that you understand the effect of using undefined and defined variables. Otherwise, you may get a number for a result instead of the algebraic expression that you anticipated.

How Undefined and Defined Variables Are Treated

When you enter an expression that contains a variable, the TI-89 Titanium treats the variable in one of two ways.

- If the variable is undefined, it is treated as an algebraic symbol.
- If the variable is defined (even if defined as 0), its value replaces the variable.

To see why this is important, suppose you want to find the first derivative of $x^3$ with respect to $x$.

- If $x$ is undefined, the result is in the form you probably expected.
• If \(x\) is defined, the result may be in a form you did not expect.

**Note:** When defining a variable, it’s a good practice to use more than one character in the name. Leave one-character names undefined for symbolic calculations.

Unless you knew that 5 had been stored to \(x\) previously, the answer 75 could be misleading.

### Determining If a Variable Is Exists

<table>
<thead>
<tr>
<th>Method:</th>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter the variable name.</td>
<td>If defined, the variable’s value is displayed.</td>
</tr>
<tr>
<td></td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td>If undefined, the variable name is displayed.</td>
</tr>
<tr>
<td>Use the <code>isVar()</code> function.</td>
<td>If defined, “true” is displayed.</td>
</tr>
<tr>
<td></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td>If undefined, “false” is displayed.</td>
</tr>
</tbody>
</table>
**Method:** Use the **getype** function.

**Example:**

If defined, the variable’s type is displayed.

| getType(x) | "NUM" |
| getType(y) | "NONE" |
| getType(z) | |

If undefined, “NONE” is displayed.

---

**Note:** Use [2nd] [VAR-LINK] to view a list of defined variables, as described in *Memory and Variable Management*. 
Deleting a Defined Variable

You can “undefine” a defined variable by deleting it.

<table>
<thead>
<tr>
<th>To delete:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>One or more specified variables</td>
<td>Use the <code>DelVar</code> function.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
DelVar x  
DelVar x,y,test,radius
```

You can also delete variables by using the VAR-LINK screen ([2nd][VAR-LINK]) as described in *Memory and Variable Management*.

<table>
<thead>
<tr>
<th>All variables of a specific type</th>
<th>Use the <code>Deltype</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note: The Deltype function deletes all variables of the specified type in all folders.</td>
<td></td>
</tr>
</tbody>
</table>

```
DelType "num"  
DelType "num"  
```

Symbolic Manipulation 238
Temporarily Overriding a Variable

By using the “with” operator (|), you can:

- Temporarily override a variable’s defined value.
- Temporarily define a value for an undefined variable.

Note: For more information about the | operator, refer to Typing the “With” Operator.

To type the “with” operator (|), press:

Symbolic Manipulation
Using Exact, Approximate, and Auto Modes

The Exact/Approx mode settings, which are described briefly in *Operating the Handheld*, directly affect the precision and accuracy with which the TI-89 Titanium calculates a result. This section describes these mode settings as they relate to symbolic manipulation.

**EXACT Setting**

When Exact/Approx = EXACT, the handheld uses exact rational arithmetic with up to 614 digits in the numerator and 614 digits in the denominator. The EXACT setting:

- Transforms irrational numbers to standard forms as much as possible without approximating them. For example, $\sqrt{12}$ transforms to $2\sqrt{3}$ and $\ln(1000)$ transforms to $3\ln(10)$.
- Converts floating-point numbers to rational numbers. For example, 0.25 transforms to $1/4$.

The functions *solve*, *cSolve*, *zeros*, *cZeros*, *factor*, *∫*, *fMin*, and *fMax* use only exact symbolic algorithms. These functions do not compute approximate solutions in the EXACT setting.

- Some equations, such as $2^{-x} = x$, have solutions that cannot all be finitely represented in terms of the functions and operators on the handheld.
• With this kind of equation, EXACT will not compute approximate solutions. For example, \(2^{-x} = x\) has an approximate solution \(x \approx 0.641186\), but it is not displayed in the EXACT setting.

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Results are exact.</td>
<td>As you use more complicated rational numbers and irrational constants, calculations can:</td>
</tr>
<tr>
<td></td>
<td>• Use more memory, which may exhaust the memory before a solution is completed.</td>
</tr>
<tr>
<td></td>
<td>• Take more computing time.</td>
</tr>
<tr>
<td></td>
<td>• Produce bulky results that are harder to comprehend than a floating-point number.</td>
</tr>
</tbody>
</table>

**APPROXIMATE Setting**

When Exact/Approx = APPROXIMATE, the handheld converts rational numbers and irrational constants to floating-point. However, there are exceptions:

• Certain built-in functions that expect one of their arguments to be an integer will convert that number to an integer if possible. For example: \(\text{d}(y(x), x, 2.0)\) transforms to \(\text{d}(y(x), x, 2)\).

• Whole-number floating-point exponents are converted to integers. For example: \(x^{2.0}\) transforms to \(x^2\) even in the APPROXIMATE setting.
Functions such as `solve` and ∫ (integrate) can use both exact symbolic and approximate numeric techniques. These functions skip all or some of their exact symbolic techniques in the APPROXIMATE setting.

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>If exact results are not needed, this might save time and/or use less memory than the EXACT setting. Approximate results are sometimes more compact and comprehensible than exact results.</td>
<td>Results with undefined variables or functions often exhibit incomplete cancellation. For example, a coefficient that should be 0 might be displayed as a small magnitude such as 1.23457E-11.</td>
</tr>
<tr>
<td>If you do not plan to use symbolic computations, approximate results are similar to familiar, traditional numeric calculators.</td>
<td>Symbolic operations such as limits and integration are less likely to give satisfying results in the APPROXIMATE setting. Approximate results are sometimes less compact and comprehensible than exact results. For example, you may prefer to see 1/7 instead of .142857.</td>
</tr>
</tbody>
</table>

**AUTO Setting**

When Exact/Approx = AUTO, the handheld uses exact rational arithmetic wherever all of the operands are rational numbers. Otherwise, floating-point arithmetic is used after
converting any rational operands to floating-point. In other words, floating-point is “infectious.” For example:

\[
\frac{1}{2} - \frac{1}{3} \text{ transforms to } \frac{1}{6}
\]
\[
\text{but}
\]
\[
0.5 - \frac{1}{3} \text{ transforms to } .16666666666667
\]

This floating-point infection does not leap over barriers such as undefined variables or between elements of lists or matrices. For example:

\[
(\frac{1}{2} - \frac{1}{3}) x + (0.5 - \frac{1}{3}) y \text{ transforms to } \frac{x}{6} + .16666666666667 y
\]
\[
\text{and}
\]
\[
\{\frac{1}{2} - \frac{1}{3}, 0.5 - \frac{1}{3}\} \text{ transforms to } \{\frac{1}{6}, .16666666666667\}
\]

In the AUTO setting, functions such as solve determine as many solutions as possible exactly, and then use approximate numerical methods if necessary to determine additional solutions. Similarly, \(\int\) (integrate) uses approximate numerical methods if appropriate where exact symbolic methods fail.

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>You see exact results when practical, and approximate numeric results when exact results are impractical. You can often control the format of a result by choosing to enter some coefficients as either rational or floating-point numbers.</td>
<td>If you are interested only in exact results, some time may be wasted seeking approximate results. If you are interested only in approximate results, some time may be wasted seeking exact results. Moreover, you might exhaust the memory seeking those exact results.</td>
</tr>
</tbody>
</table>
**Automatic Simplification**

When you type an expression on the entry line and press [ENTER], the TI-89 Titanium automatically simplifies the expression according to its default simplification rules.

**Default Simplification Rules**

All of the following rules are applied automatically. You do not see intermediate results.

- If a variable has a defined value, that value replaces the variable.
  
  If the variable is defined in terms of another variable, the variable is replaced with its “lowest level” value (called infinite lookup).

Default simplification does not modify variables that use path names to indicate a folder. For example, `x+class\x` does not simplify to `2x`.

**Note:** For information about folders, refer to the *Calculator Home Screen* module.

- For functions:
  - The arguments are simplified. (Some built-in functions delay simplification of some of their arguments.)
  - If the function is a built-in or user-defined function, the function definition is applied to the simplified arguments. Then the functional form is replaced with this result.
Symbolic Manipulation

- Numeric subexpressions are combined.
- Products and sums are sorted into order.

Products and sums involving undefined variables are sorted according to the first letter of the variable name.

- Undefined variables r through z are assumed to be true variables, and are placed in alphabetical order at the beginning of a sum.
- Undefined variables a through q are assumed to represent constants, and are placed in alphabetical order at the end of a sum (but before numbers).

- Similar factors and similar terms are collected.
- Identities involving zeros and ones are exploited.

This floating-point number causes numeric results to be shown as floating-point.

If a floating-point whole number is entered as an exponent, it is treated as an integer (and does not produce a floating-point result).
• Polynomial greatest common divisors are canceled.

• Polynomials are expanded unless no key cancellation can occur.

• Common denominators are formed unless no key cancellation can occur.

• Functional identities are exploited. For example:
  \[ \ln(2x) = \ln(2) + \ln(x) \]
  and
  \[ \sin(x)^2 + \cos(x)^2 = 1 \]

**How Long Is the Simplification Process?**

Depending on the complexity of an entry, result, or intermediate expression, it can take a long time to expand an expression and cancel common divisors as necessary for simplification.

To interrupt a simplification process that is taking too long, press \[ \text{ON} \]. You can then try simplifying only a portion of the expression. (Auto-paste the entire expression on the entry line, and then delete the unwanted parts.)
Delayed Simplification for Certain Built-In Functions

Usually, variables are automatically simplified to their lowest possible level before they are passed to a function. For certain functions, however, complete simplification is delayed until after the function is performed.

Functions that Use Delayed Simplification

Functions that use delayed simplification have a required var argument that performs the function with respect to a variable. These functions have at least two arguments with the general form:

\[
\text{function}(\text{expression}, \text{var} [, \ldots])
\]

Note: Not all functions that use a var argument use delayed simplification.

For example:

\[
\text{solve}(x^2-x-2=0,x)
\]
\[
\text{d}(x^2-x-2,x)
\]
\[
\text{int}(x^2-x-2,x)
\]
\[
\text{limit}(x^2-x-2,x,5)
\]

For a function that uses delayed simplification:

1. The var variable is simplified to the lowest level at which it remains a variable (even if it could be further simplified to a non-variable value).
2. The function is performed using the variable.
3. If var can be further simplified, that value is then substituted into the result.
Note: You may or may not want to define a numeric value for \( \text{var} \), depending on the situation.

For example:

\[
\frac{d}{dx}(x^3) = 3x^2
\]

\[
5 \rightarrow x \quad 5
\]

\[
\frac{d}{dx}(x^3) = 75
\]

\[
\text{MAIN} \quad \text{RAD RUTE} \quad \text{FUNC} \quad 2/20
\]

\[
\frac{d}{dx}(x^3) = 3x^2
\]

\[
5 \rightarrow t \quad 5
\]

\[
\frac{d}{dx}(t^3) = 75
\]

\[
\text{MAIN} \quad \text{RAD RUTE} \quad \text{FUNC} \quad 2/20
\]

Note: The example to the right finds the derivative of \( x^3 \) at \( x=5 \). If \( x^3 \) was initially simplified to 75, you would find the derivative of 75, which is not what you want.

\[
\frac{d}{dx}(x^3) = 3x^2
\]

\[
5 \rightarrow t \quad 5
\]

\[
\frac{d}{dx}(t^3) = 75
\]

\[
\text{MAIN} \quad \text{RAD RUTE} \quad \text{FUNC} \quad 2/20
\]

\[
\frac{d}{dx}(t^3) = 3t^2
\]

x cannot be simplified.

x is not simplified. The function uses \( x^3 \), and then substitutes 5 for \( x \).

x is simplified to \( t \). The function uses \( t^3 \).

x is simplified to \( t \). The function uses \( t^3 \), and then substitutes 5 for \( t \).
Substituting Values and Setting Constraints

The “with” operator ( | ) lets you temporarily substitute values into an expression or specify domain constraints.

Typing the “With” Operator

To type the “with” operator ( | ), press:

Substituting for a Variable

For every occurrence of a specified variable, you can substitute a numeric value or an expression.

To substitute for multiple variables at the same time, use the Boolean and operator.
Substituting for a Simple Expression

For every occurrence of a simple expression, you can substitute a variable, numeric value, or another expression.

Substituting Complex Values

You can substitute complex values just as you would for other values.

Note: \( \text{acos}(x) \) is different from \( a \cdot \cos(x) \).

Substituting Complex Values

All undefined variables are treated as real numbers in symbolic calculations. To perform complex symbolic analysis, you must define a complex variable. For example:

\[ x + yi \rightarrow z \]

Then you can use \( z \) as a complex variable. You can also use \( z_\_ \). For more information see the _ (underscore) topic in the Technical Reference module.
Note:

- For an overview of complex numbers, refer to the *Technical Reference* module.
- To get the complex $i$, press `2nd [i]`. Do not simply type the letter $i$ on the keyboard.

**Be Aware of the Limitations of Substitutions**

- Substitution occurs only where there is an exact match for the substitution.

Only $x^2$ was replaced, not $x^4$.

Define the substitution in simpler terms for a more complete substitution.
• Infinite recursions can occur when you define a substitution variable in terms of itself.

\[
\text{Substitutes } \sin(x+1), \sin(x+1+1), \sin(x+1+1+1), \text{ etc}
\]

\[
\sin(x) | x = x + 1
\]

When you enter a substitution that causes an infinite recursion:

- An error message is displayed.

- When you press [ESC], an error is shown in the history area.

• Internally, an expression is sorted according to the automatic simplification rules. Therefore, products and sums may not match the order in which you entered them.

- As a general rule, you should substitute for a single variable.
Substituting for more general expressions (either $m \cdot c^2 = e$ or $c^2 \cdot m = e$) may not work as you anticipate.

**Note:** Use the `solve` function to help determine the single-variable substitution.

### Specifying Domain Constraints

Many identities and transformations are valid for only a particular domain. For example:

\[
\ln(x \cdot y) = \ln(x) + \ln(y) \quad \text{only if } x \text{ and/or } y \text{ is not negative}
\]

\[
\sin^{-1}(\sin(\theta)) = \theta \quad \text{only if } \theta \geq -\pi/2 \text{ and } \theta \leq \pi/2 \text{ radians}
\]

Use the “*with*” operator to specify the domain constraint.

Because $\ln(x \cdot y) = \ln(x) + \ln(y)$ is not always valid, the logarithms are not combined.

With a constraint, the identity is valid and the expression is simplified.
**Note:** Enter \( \ln(x*y) \) instead of \( \ln(xy) \); otherwise, \( xy \) is interpreted as a single variable named \( xy \).

Because \( \sin^{-1}(\sin(\theta)) = \theta \) is not always valid, the expression is not simplified.

With a constraint, the expression can be simplified.

**Note:** For \( \geq \) or \( \leq \), press \( \bullet [>] \) or \( \bullet [<] \). You can also use \( \text{2nd [MATH]} \) 8 or \( \text{2nd [CHAR]} \) 2 to select them from a menu.

### Using Substitutions vs. Defining a Variable

In many cases, you can achieve the same effect as a substitution by defining the variable.

However, substitution is preferable for most cases because the variable is defined only for the current calculation and does not accidentally affect later calculations.

Substituting \( x=1 \) does not affect the next calculation.
Storing $1 \rightarrow x$ affects the subsequent calculations.

**Caution:** After $x$ is defined, it can affect all calculations that involve $x$ (until you delete $x$).

## Overview of the Algebra Menu

You can use the F2 Algebra toolbar menu to select the most commonly used algebraic functions.

### The Algebra Menu

From the Home screen, press F2 to display:

This menu is also available from the MATH menu. Press 2nd [MATH] and then select 9:Algebra.
Note: For a complete description of each function and its syntax, refer to the *Technical Reference* module.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>solve</td>
<td>Solves an equation for a specified variable. This returns real solutions only, regardless of the Complex Format mode setting. Displays answers with &quot;and&quot; and &quot;or&quot; connecting solutions. (For complex solutions, select A:Complex from the Algebra menu.)</td>
</tr>
<tr>
<td>factor</td>
<td>Factors an expression with respect to all its variables or with respect to only a specified variable.</td>
</tr>
<tr>
<td>expand</td>
<td>Expands an expression with respect to all its variables or with respect to only a specified variable.</td>
</tr>
<tr>
<td>zeros</td>
<td>Determines the values of a specified variable that make an expression equal to zero. Displays in a list.</td>
</tr>
<tr>
<td>approx</td>
<td>Evaluates an expression using floating-point arithmetic, where possible. This is equivalent to using [MODE] to set Exact/Approx = APPROXIMATE (or using [♦] [ENTER] to evaluate an expression).</td>
</tr>
<tr>
<td>comDenom</td>
<td>Calculates a common denominator for all terms in an expression and transforms the expression into a reduced ratio of a numerator and denominator.</td>
</tr>
<tr>
<td>propFrac</td>
<td>Returns an expression as a proper fraction expression.</td>
</tr>
<tr>
<td>nSolve</td>
<td>Calculates a single solution for an equation as a floating-point number (as opposed to solve, which may display several solutions in a rational or symbolic form).</td>
</tr>
<tr>
<td>Menu Item</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Trig</strong></td>
<td>Displays the submenu:</td>
</tr>
<tr>
<td><strong>tExpand</strong> — Expands trig expressions with angle sums and multiple angles.</td>
<td></td>
</tr>
<tr>
<td><strong>TCollect</strong> — Collects the products of integer powers of trig functions into angle sums and multiple angles. <strong>tCollect</strong> is the opposite of <strong>tExpand</strong>.</td>
<td></td>
</tr>
<tr>
<td><strong>Complex</strong></td>
<td>Displays the submenu:</td>
</tr>
<tr>
<td><strong>cSolve</strong></td>
<td>These are the same as <strong>solve</strong>, <strong>factor</strong>, and <strong>zeros</strong>; but they also compute complex results.</td>
</tr>
<tr>
<td><strong>Extract</strong></td>
<td>Displays the submenu:</td>
</tr>
<tr>
<td><strong>getNum</strong> — Applies <strong>comDenom</strong> and then returns the resulting numerator.</td>
<td></td>
</tr>
<tr>
<td><strong>getDenom</strong> — Applies <strong>comDenom</strong> and then returns the resulting denominator.</td>
<td></td>
</tr>
<tr>
<td><strong>left</strong> — Returns the left-hand side of an equation or inequality.</td>
<td></td>
</tr>
</tbody>
</table>
Note: The left and right functions are also used to return a specified number of elements or characters from the left or right side of a list or character string.

**Common Algebraic Operations**

This section gives examples for some of the functions available from the [F2] Algebra toolbar menu. For complete information about any function, refer to the *Technical Reference* module. Some algebraic operations do not require a special function.

**Adding or Dividing Polynomials**

You can add or divide polynomials directly, without using a special function.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>right</strong></td>
<td>Returns the right-hand side of an equation or inequality.</td>
</tr>
</tbody>
</table>
Factoring and Expanding Polynomials

Use the **factor** ([F2] 2) and **expand** ([F2] 3) functions.

\[
\text{factor(expression [,var])}
\]

for factoring with respect to a variable

\[
\text{expand(expression [,var])}
\]

for partial expansion with respect to a variable

Factor \(x^5 - 1\). Then expand the result.

Notice that **factor** and **expand** perform opposite operations.

Finding Prime Factors of a Number

The **factor** ([F2] 2) function lets you do more than simply factor an algebraic polynomial.

You can find prime factors of a rational number (either an integer or a ratio of integers).
Finding Partial Expansions

With the \texttt{expand ([F2] 3)} function’s optional \texttt{var} value, you can do a partial expansion that collects similar powers of a variable.

Do a full expansion of \((x^2-x)(y^2-y)\) with respect to all variables.

Then do a partial expansion with respect to \(x\).

\begin{itemize}
  \item \texttt{expand([(x^2-x)\cdot(y^2-y)])}
  \item \texttt{expand([(x^2-x)\cdot(y^2-y), x])}
  \item \texttt{expand((x^2-x)\cdot(y^2-y), x)}
\end{itemize}

Solving an Equation

Use the \texttt{solve ([F2] 1)} function to solve an equation for a specified variable.

\texttt{solve(equation, var)}

Solve \(x + y - 5 = 2x - 5y\) for \(x\).

Notice that solve displays only the final result.
To see intermediate results, you can manually solve the equation step-by-step.

\[
x + y - 5 \equiv 2x - 5y
\]

\[
\begin{align*}
\text{\(\boxed{-2x}\)} & \quad \text{\(\boxed{-y}\)} & \quad \text{\(\boxed{+5}\)} & \quad \text{\(\boxed{\times (-1)}\)}
\end{align*}
\]

**Note:** An operation such as \(\boxed{-2 \times}\) subtracts 2x from both sides.

**Solving a System of Linear Equations**

Consider a set of two equations with two unknowns:

\[
\begin{align*}
2x - 3y &= 4 \\
-x + 7y &= -12
\end{align*}
\]

To solve this system of equations, use any of the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the <code>solve</code> function for a one-step solution.</td>
<td><code>solve(2x-3y=4 \text{ and } -x+7y=-12,\{x,y\})</code></td>
</tr>
<tr>
<td>Use the <code>solve</code> function with substitution ((</td>
<td>)) for step-by-step manipulation.</td>
</tr>
</tbody>
</table>
### Method | Example
--- | ---
Use the `simult` function with a matrix. | Enter the coefficients as a matrix and the results as a constant column matrix.

```
\[ \text{simult} \begin{bmatrix} 2 & -3 \\ -1 & 7 \end{bmatrix}, \begin{bmatrix} 4 \\ -12 \end{bmatrix} \]
```

```
\text{ult}(12, -3; -1, 7, [4; -12])
```

Use the `rref` function with a matrix. | Enter the coefficients as an augmented matrix.

```
\[ \text{rref} \begin{bmatrix} 2 & -3 & 4 \\ -1 & 7 & -12 \end{bmatrix} \]
```

```
\text{rref}(12, -3, 4; -1, 7, -12)
```

---

**Note:** The `simult` and `rref` matrix functions are not on the `F2 Algebra` menu. Use `2nd [MATH] 4` or the `Catalog`. 

---

**Symbolic Manipulation**
Finding the Zeros of an Expression

Use the `zeros` ([F2] 4) function.

\[
\text{zeros(expression, var)}
\]

Use the expression \(x\sin(x) + \cos(x)\).

Find the zeros with respect to \(x\) in the interval \(0 \leq x \leq 3\).

**Note:** For \(\geq\) or \(\leq\), type \(\geq\) [>] or \(\leq\) [<]. You can also use [2nd] [MATH] 8 or [2nd] [CHAR] 2 to select them from a menu.
Finding Proper Fractions and Common Denominators

Use the propFrac (F2 7) and comDenom (F2 6) functions.

\[ \text{propFrac}(\text{rational expression}[,\text{var}]) \]
- for proper fractions with respect to a variable

\[ \text{comDenom}(\text{expression}[,\text{var}]) \]
- for common denominators that collect similar powers of this variable

Find a proper fraction for the expression \( \frac{x^4 - 2x^2 + x}{2x^2 + x + 4} \).

Then transform the answer into a ratio of a fully expanded numerator and a fully expanded denominator.

Notice that propFrac and comDenom perform opposite operations.

**Note:** You can use comDenom with an expression, list, or matrix.

If you do this example on your handheld, the propFrac function scrolls off the top of the screen.

In this example:
• \( \frac{31x + 60}{8} \) is the remainder of \( x^4 - 2x^2 + x \) divided by \( 2x^2 + x + 4 \).

• \( \frac{x^2}{2} - \frac{x}{4} - 15/8 \) is the quotient.

### Overview of the Calc Menu

You can use the F3 Calc toolbar menu to select commonly used calculus functions.

### The Calc Menu

From the Home screen, press F3 to display:

```
<table>
<thead>
<tr>
<th>1: d</th>
<th>differentiate</th>
</tr>
</thead>
<tbody>
<tr>
<td>2: I</td>
<td>Integrate</td>
</tr>
<tr>
<td>3: s</td>
<td>sum</td>
</tr>
<tr>
<td>4: P</td>
<td>Product</td>
</tr>
<tr>
<td>5: fM</td>
<td>fMin</td>
</tr>
<tr>
<td>6: fMx</td>
<td>fMax</td>
</tr>
<tr>
<td>7: var</td>
<td>arcLen</td>
</tr>
</tbody>
</table>
```

This menu is also available from the MATH menu. Press 2nd [MATH] and then select A:Calculus.

**Note:** For a complete description of each function and its syntax, refer to the Technical Reference module.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d differentiate</td>
<td>Differentiates an expression with respect to a specified variable.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>∫ integrate</td>
<td>Integrates an expression with respect to a specified variable.</td>
</tr>
<tr>
<td>limit</td>
<td>Calculates the limit of an expression with respect to a specified variable.</td>
</tr>
<tr>
<td>Σ sum</td>
<td>Evaluates an expression at discrete variable values within a range and then calculates the sum.</td>
</tr>
<tr>
<td>Π product</td>
<td>Evaluates an expression at discrete variable values within a range and then calculates the product.</td>
</tr>
<tr>
<td>fMin</td>
<td>Finds candidate values of a specified variable that minimize an expression.</td>
</tr>
<tr>
<td>fMax</td>
<td>Finds candidate values of a specified variable that maximize an expression.</td>
</tr>
<tr>
<td>arcLen</td>
<td>Returns the arc length of an expression with respect to a specified variable.</td>
</tr>
<tr>
<td>taylor</td>
<td>Calculates a Taylor polynomial approximation to an expression with respect to a specified variable.</td>
</tr>
<tr>
<td>nDeriv</td>
<td>Calculates the numerical derivative of an expression with respect to a specified variable.</td>
</tr>
<tr>
<td>nInt</td>
<td>Calculates an integral as a floating-point number using quadrature (an approximation using weighted sums of integrand values).</td>
</tr>
<tr>
<td>deSolve</td>
<td>Symbolically solves many 1st and 2nd order differential equations, with or without initial conditions.</td>
</tr>
<tr>
<td>impDif</td>
<td>Computes implicit derivatives for equations in two variables in which one variable is defined implicitly in terms of another.</td>
</tr>
</tbody>
</table>
Note: The $d$ symbol for differentiate is a special symbol. It is not the same as typing the letter D on the keyboard. Use $[\text{F3 1]}$ or $[2\text{nd }d]$. 

**Common Calculus Operations**

This section gives examples for some of the functions available from the $[\text{F3 Calc}]$ toolbar menu. For complete information about any calculus function, refer to the *Technical Reference* module.

**Integrating and Differentiating**

Use the $\int \text{integrate (F3 2)}$ and $d \text{differentiate (F3 1)}$ functions.

\[
\int (\text{expression, var [,low] [,up]})
\]

lets you specify limits or a constant of integration

\[
d (\text{expression, var [,order]})
\]

Integrate $x^2 \cdot \sin(x)$ with respect to $x$.

Differentiate the answer with respect to $x$.

To get $d$, use $[\text{F3 1}]$ or $[2\text{nd }d]$. Do not simply type the letter D on the keyboard.
**Note:** You can integrate an expression only; you can differentiate an expression, list, or matrix.

### Finding a Limit

Use the `limit` (F3 3) function.

\[
\text{limit}(\text{expression, \ var, \ point [,direction]}\])
\]

- negative number = from left
- positive number= from right
- omitted number or 0 = both

Find the limit of \(\frac{\sin(3x)}{x}\) as \(x\) approaches 0.

\[
\lim_{x \to 0} \left( \frac{\sin(3x)}{x} \right)
\]

**Note:** You can find a limit for an expression, list, or matrix.
Finding a Taylor Polynomial

Use the `taylor` ([F3] 9) function.

\[
taylor(expression, \text{var}, order [, point])
\]

if omitted, expansion point is 0

Find a 6th order Taylor polynomial for \(\sin(x)\) with respect to \(x\).

Store the answer as a user-defined function named \(y1(x)\).

Then graph \(\sin(x)\) and the Taylor polynomial.

Important: Degree-mode scaling by \(\pi/180\) may cause calculus application results to appear in a different form.

User-Defined Functions and Symbolic Manipulation

You can use a user-defined function as an argument for the TI-89 Titanium ’s built-in algebra and calculus functions.
For Information about Creating a User-Defined Function

Refer to:

• “Creating and Evaluating User-Defined Functions” in the Calculator Home Screen module.
• “Graphing a Function Defined on the Home Screen” and “Graphing a Piecewise Defined Function” in the Calculator Home Screen module.
• “Overview of Entering a Function” in the Programming module.

Undefined Functions

You can use functions such as \( f(x) \), \( g(t) \), \( r(\theta) \), etc., that have not been assigned a definition. These “undefined” functions yield symbolic results. For example:

Use \texttt{DelVar} to ensure that \( f(x) \) and \( g(x) \) are not defined.

Then find the derivative of \( f(x) \, g(x) \) with respect to \( x \).

\[
\frac{d}{dx}(f(x) \cdot g(x)) = \frac{d}{dx}(f(x)) \cdot g(x) + f(x) \cdot \frac{d}{dx}(g(x))
\]

\[
\frac{d}{dx}(f(x) \cdot g(x) + f(x) \cdot g(x))
\]

\[
\text{Main} \quad \text{Graph Auto} \quad \text{Func} \quad 2, \text{dr}
\]

\textbf{Note:} To select \texttt{d} from the Calc toolbar menu, press \texttt{F3} 1 (or press \texttt{2nd} [\texttt{d}] on the keyboard).

Single-Statement Functions

You can use user-defined functions consisting of a single expression. For example:
• Use \texttt{STO} to create a user-defined secant function, where:

\[
\sec x = \frac{1}{\cos x}
\]

Then find the limit of \( \sec(x) \) as \( x \) approaches \( \pi/4 \).

\textbf{Note:} To select \texttt{limit} from the Calc toolbar menu, press \texttt{F3} 3.

• Use \texttt{Define} to create a user-defined function \( h(x) \), where:

\[
h(x) = \int_{0}^{x} \frac{\sin t}{t} dt
\]

Then find a 5th order Taylor polynomial for \( h(x) \) with respect to \( x \).

\textbf{Note:} To select \( \int \) from the \texttt{Calc} toolbar menu, press \texttt{F3} 2 (or press \texttt{2nd} [\( \int \)] on the keyboard). To select \texttt{taylor}, press \texttt{F3} 9.

\textbf{Multi-Statement vs. Single-Statement Functions}

Multi-statement user-defined functions should be used as an argument for numeric functions (such as \texttt{nDeriv} and \texttt{nInt}) only.
In some cases, you may be able to create an equivalent single-statement function. For example, consider a piecewise function with two pieces.

### Table: Use expression

<table>
<thead>
<tr>
<th>When:</th>
<th>Use expression:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x &lt; 0 )</td>
<td>(-x)</td>
</tr>
<tr>
<td>( x \geq 0 )</td>
<td>(5\cos(x))</td>
</tr>
</tbody>
</table>

- If you were to create a multi-statement user-defined function with the form:

  ```
  Func
    If x<0 Then
      Return \( \text{expr} \)
    Else
      Return \(5\cos(x)\)
  EndIf
  EndFunc
  
  Define
  y1(x)=Func:If x<0
  Then: ... :EndFunc
  
  Then numerically integrate \( y_1(x) \) with respect to \( x \).

**Note:** To select \( \text{nInt} \) from the Calc toolbar menu, press [F3] B:nInt.
• Create an equivalent single-statement user-defined function.

Use the TI-89 Titanium’s built-in **when** function.

Then integrate \( y_1(x) \) with respect to \( x \).

**Note:** To select \( \int \) from the Calc toolbar menu, press \( \text{F3} \ 2 \) (or press \( \text{2nd} \ [\int] \) on the keyboard).

Define
\[
y_1(x) = \begin{cases} 
-x, & x < 0 \\
5 \cos(x), & x \geq 0 
\end{cases}
\]

Press \( \text{F3} \) \( \text{F5} \) \( \text{F3} \) to view and delete variables and/or Flash applications.

\[
\int_0^1 y_1(x) \, dx = 5 \cdot \sin(1)
\]

\[
\int_0^1 y_1(x) \, dx = 4.28735
\]

If You Get an Out-of-Memory Error

The TI-89 Titanium stores intermediate results in memory and then deletes them when the calculation is complete. Depending on the complexity of the calculation, the handheld may run out of memory before a result can be calculated.

**Freeing Up Memory**

• Delete unneeded variables and/or Flash applications, particularly large-sized ones.
  
  - Use \( \text{2nd} \ [\text{VAR-LINK}] \) as described in *Memory and Variable Management* to view and delete variables and/or Flash applications.

• On the Home screen:
- Clear the history area ([F1] 8) or delete unneeded history pairs.
- You can also use [F1] 9 to reduce the number of history pairs that will be saved.

- **Use [MODE]** to set Exact/Approx = APPROXIMATE. (For results that have a large number of digits, this uses less memory than **AUTO** or EXACT. For results that have only a few digits, this uses more memory.)

**Simplifying Problems**

- Split the problem into parts.
  - Split \(\text{solve}(a*b=0, \text{var})\) into \(\text{solve}(a=0, \text{var})\) and \(\text{solve}(b=0, \text{var})\). Solve each part and combine the results.

- If several undefined variables occur only in a certain combination, replace that combination with a single variable.
  - If \(m\) and \(c\) occur only as \(m*c^2\), substitute \(e\) for \(m*c^2\).
  - In the expression \(\frac{(a + b)^2 + \sqrt{(a + b)^2}}{1 - (a + b)^2}\), substitute \(c\) for \((a+b)\) and use \(\frac{c^2 + \sqrt{c^2}}{1 - c^2}\). In the solution, replace \(c\) with \((a+b)\).

- For expressions combined over a common denominator, replace sums in denominators with unique new undefined variables.
  - In the expression \(\frac{x}{\sqrt{a^2 + b^2} + c} + \frac{y}{\sqrt{a^2 + b^2} + c}\), substitute \(d\) for \(\sqrt{a^2 + b^2} + c\) and are \(\frac{x}{d} + \frac{y}{d}\). In the solution, replace \(d\) with \(\sqrt{a^2 + b^2} + c\).

- Substitute known numeric values for undefined variables at an earlier stage, particularly if they are simple integers or fractions.
• Reformulate a problem to avoid fractional powers.
• Omit relatively small terms to find an approximation.

**Special Constants Used in Symbolic Manipulation**

The result of a calculation may include one of the special constants described in this section. In some cases, you may also need to enter a constant as part of your entry.

**true, false**

These indicate the result of an identity or a Boolean expression.

- \( x = x \) is true for any value of \( x \).
- \( 5 \leq x : x < 3 \) is false.
- \( 5 < 3 \) is false.
This notation indicates an “arbitrary integer” that represents any integer.

When an arbitrary integer occurs multiple times in the same session, each occurrence is numbered consecutively. After it reaches 255, arbitrary integer consecutive numbering restarts at @n0. Use Clean Up 2:NewProb to reset to @n1.

Note: For @, press: ▼ STO►

A solution is at every integer multiple of π.

Both @n and @n2 represent any arbitrary integer, but this notation identifies separate arbitrary integers.
This notation indicates an “arbitrary constant” that represents any integer.

When an arbitrary constant occurs multiple times in the same session, each occurrence is numbered consecutively. After it reaches 255, arbitrary integer consecutive numbering restarts at @0. Use Clean Up 2:NewProb to reset to @1.

Note: For @, press:  [STO↓]

∞, e

∞ represents infinity, and e represents the constant 2.71828... (base of the natural logarithms).

These constants are often used in entries as well as results.

Notes:
For ∞, press:

For e, press:

Symbolic Manipulation
This indicates that the result is undefined.

Mathematically undefined

±∞ (undetermined sign)

Non-unique limit
Entering Constants or Units

You can use a menu to select from a list of available constants and units, or you can type them directly from the keyboard.

From a Menu

The following shows how to select a unit, but you can use the same general procedure to select a constant.

From the Home screen:

1. Type the value or expression.

2. Display the UNITS dialog box. Press: 2nd [UNITS]

3. Use  and  to move the cursor to the applicable category.

Note: Use 2nd  and 2nd  to scroll one page at a time through the categories.
4. To select the highlighted (default) unit, press ENTER.
   – or –
   To select a different unit from the category, press ◀. Then highlight the applicable unit, and press ENTER.

   **Note:** If you created a user-defined unit for an existing category, it is listed in the menu.

   The selected unit is placed in the entry line. Constant and unit names always begin with an underscore ( _ ).

   ![Image of unit selection screen]

   You can also move the cursor by typing the first letter of a unit.

   6.3_pF

### From the Keyboard

If you know the abbreviation that the TI-89 Titanium uses for a particular constant or unit, you can type it directly from the keyboard. For example:

   **256_m**

   • The first character must be an underscore ( _ ). For _, press: 
     ◀ [−]

   • A space or a multiplication symbol (*) before the underscore is optional. For example, 256_m, 256 _m, and 256* _m are equivalent.

   • However, if you are adding units to a variable, you must put a space or * before the underscore. For example, x_m is treated as a variable, not as x with a unit.

   **Note:** You can type units in either uppercase or lowercase characters.
Combining Multiple Units

You may need to combine two or more units from different categories.

For example, suppose you want to enter a velocity in meters per second. In the UNITS dialog box, however, the Velocity category does not contain this unit.

You can enter meters per second by combining \texttt{\_m} and \texttt{\_s} from the Length and Time categories, respectively.

\begin{equation}
3 \times 9.8 \text{ m/} \text{s}
\end{equation}

Combine the units \texttt{\_m} and \texttt{\_s}. There is no pre-defined m/\text{s} unit.

\textbf{Note}: Create a user-defined unit for frequently used combinations.
Using Parentheses with Units in a Calculation

In a calculation, you may need to use parentheses ( ) to group a value and its units so that they are evaluated properly. This is particularly true for division problems. For example:

<table>
<thead>
<tr>
<th>To calculate:</th>
<th>Enter:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{100 \text{ m}}{2 \text{ s}} )</td>
<td>( 100 \text{ m} / (2 \text{ s}) ) ( 50 \cdot \frac{\text{m}}{\text{s}} )</td>
</tr>
</tbody>
</table>

You must use parentheses for \((2 \text{ s})\). This is important for division.

If you omit the parentheses, you will get unexpected units. For example:

\[ 100 \text{ m} / 2 \text{ s} \quad 50. \cdot \text{m} \cdot \text{s} \]

**Note:** If you have any doubt about how a value and its units will be evaluated, group them within parentheses ( ).

Here’s why you get unexpected units if you do not use parentheses. In a calculation, a unit is treated similar to a variable. For example: \( 100 \text{ m} \) is treated as \( 100 \cdot \text{m} \) and \( 2 \text{ s} \) is treated as \( 2 \cdot \text{s} \). Without parentheses, the entry is calculated as:

\[ 100 \cdot \text{m} / 2 \cdot \text{s} = \frac{100 \cdot \text{m}}{2} \cdot \text{s} = 50. \cdot \text{m} \cdot \text{s}. \]
Converting from One Unit to Another

You can convert from one unit to another in the same category, including any user-defined units.

For All Units Except Temperature

If you use a unit in a calculation, it is converted and displayed automatically in the current default unit for that category, unless you use the conversion operator as described later. The following examples assume that your default units are set to the SI system of metric units.

Notes:

- Refer to the list of pre-defined units.
- From the UNITS dialog box, you can select available units from a menu.

To multiply 20 times 6 kilometers.

\[ 20 \times 6_{\text{km}} \]

Shown in the default unit for Length, \( _{\text{m}} \) in SI system).
If you want to convert to a unit other than the default, use the \( \Rightarrow \) conversion operator.

\[ \text{expression\_unit1} \Rightarrow \text{unit2} \]

- For \( \Rightarrow \), press \( \text{2nd} \) [\( \Rightarrow \)].

To convert 4 light years to kilometers:

\[ 4\text{\_ltyr} \Rightarrow \text{\_km} \]

To convert 186000 miles/second to kilometers/hour:

\[ 186000\text{\_mi/\_s} \Rightarrow \text{\_km/\_hr} \]
If an expression uses a combination of units, you can specify a conversion for some of the units only. Any units for which you do not specify a conversion will be displayed according to your defaults.

To convert 186000 miles/second from miles to kilometers:

\[ 186000 \text{ mi/s} \rightarrow \text{km} \]

To convert 186000 miles/second from seconds to hours:

\[ 186000 \text{ mi/s} \rightarrow 1/\text{hr} \]

Because a Time conversion is not specified, it is shown in its default unit (\(\text{s}\) in this example).

Because a Length conversion is not specified, it is shown in its default unit (\(\text{m}\) in this example).

To enter meters per second squared:

\[ 27 \text{ m/s}^2 \]

To convert meters per second squared from seconds to hours:

\[ 27 \text{ m/s}^2 \rightarrow 1/\text{hr}^2 \]
For Temperature Values

To convert a temperature value, you must use `tmpCnv()` instead of the operator.

\[
\text{tmpCnv(} \text{expression}_\circ \text{tempUnit1, } \circ \text{tempUnit2})
\]

For °, press \text{2nd} [°].

For example, to convert 100 °C to °F:

\[
\text{tmpCnv(} 100_\circ \text{c, } _\circ \text{f})
\]

\[
\begin{array}{|c|}
\hline
\text{tmpCnv(}100_\circ \text{c, } _\circ \text{f}) \rightarrow \text{212}_\circ \text{f} \\
\text{tmpCnv(}100_\circ \text{c, } _\circ \text{f}) \rightarrow \text{212}_\circ \text{f} \\
\text{MAIN} \rightarrow \text{RAD AUTO} \rightarrow \text{FUNC} \rightarrow \text{1/20} \\
\hline
\end{array}
\]

\[
\begin{array}{c}
0 \quad 100 \quad ^\circ \text{C} \\
32 \quad 212 \quad ^\circ \text{F}
\end{array}
\]
For Temperature Ranges

To convert a temperature range (the difference between two temperature values), use \( \Delta \text{tmpCnv}() \).

\[ \Delta \text{tmpCnv}(\text{expression} \_\text{\degree tempUnit1}, \_\text{\degree tempUnit2}) \]

For example, to convert a 100\_\degree C range to its equivalent range in \_\degree F:

\[ \Delta \text{tmpCnv}(100\_\degree C, \_\degree F) \]

Note: For \( \Delta \), press:
\[ \blacklozenge \ lacklozenge \ [D] \]

Setting the Default Units for Displayed Results

All results involving units are displayed in the default unit for that category. For example, if the default unit for Length is \_m, any length result is displayed in meters (even if you entered \_km or \_ft in the calculation).
If You’re Using the SI or ENG/US System

The SI and ENG/US systems of measurement (set from Page 3 of the MODE screen) use built-in default units, which you cannot change.

The default units for these systems are available.

Setting Custom Defaults

To set custom defaults:

1. Press \text{MODE} \ F3 \ 3 to set Unit System = CUSTOM.

2. Press \text{ } to highlight SET DEFAULTS.

3. Press \text{ } to display the CUSTOM UNIT DEFAULTS dialog box.
4. For each category, you can highlight its default, press ‹, and select a unit from the list.

5. Press ENTER twice to save your changes and exit the MODE screen.

Notes:

- You can also use setUnits( ) or getUnits( ) to set or return information about default units. Refer to the Technical Reference module.

- When the CUSTOM UNIT DEFAULTS dialog box first appears, it shows the current default units.

What is the NONE Default?

Many categories let you select NONE as the default unit.

This means that results in that category are displayed in the default units of its components.

For example, Area = Length², so Length is the component of Area.
• If the defaults are **Area = _acre** and **Length = _m** (meters), area results are shown with **_acre** units.

• If you set **Area = NONE**, area results are shown with **_m^2** units.

**Note:** NONE is not available for base categories such as **Length** and **Mass** that have no components.

### Creating Your Own User-Defined Units

In any category, you can expand the list of available units by defining a new unit in terms of one or more pre-defined units. You can also use “standalone” units.

### Why Use Your Own Units?

Some example reasons to create a unit are:

• You want to enter length values in dekameters. Define **10_m** as a new unit named **_dm**.

• Instead of entering **_m/_s^2** as an acceleration unit, you define that combination of units as a single unit named **_ms2**.

• You want to calculate how many times someone blinks. You can use **_blinks** as a valid unit without defining it. This “standalone” unit is treated similar to a variable that is not defined. For instance, **3_blinks** is treated the same as **3a**.

**Note:** If you create a user-defined unit for an existing category, you can select it from the UNITS dialog box menu. But you cannot use **MODE** to select the unit as a default for displayed results.
**Rules for User-Defined Unit Names**

The naming rules for units are similar to variables.

- Can have up to 8 characters.
- First character must be an underscore. For _, press: 
  
- Second character can be any valid variable name character except _ or a digit. For example, _9f is not valid.
- Remaining characters (up to 6) can be any valid variable name character except an underscore.

**Defining a Unit**

Define a unit the same way you store to a variable.
For example, to define a dekameter unit:

\[ 10\,\text{m} \to _{\text{dm}} \]

To define an acceleration unit:

\[ _{\text{m}}/\text{s}^2 \to _{\text{ms}^2} \]

To calculate 195 blinks in 5 minutes as _blinks/_min:

\[ 195\,\text{blinks}/(5\,\text{min}) \]

Notes:

- User-defined units are displayed in lowercase characters, regardless of the case you use to define them.
- User-defined units such as _dm are stored as variables. You can delete them the same as you would any variable.
List of Pre-Defined Constants and Units

This section lists the pre-defined constants and units by category. You can select any of these from the UNITS dialog box. If you use [MODE] to set default units, note that categories with only one defined unit are not listed.

Defaults for SI and ENG/US

The SI and ENG/US systems of measurement use built-in default units. In this section, the built-in defaults are indicated by (SI) and (ENG/US). In some categories, both systems use the same default.

Some categories do not have default units.

Constants

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_c</td>
<td>speed of light 2.99792458E8 m/s</td>
</tr>
<tr>
<td>_Cc</td>
<td>coulomb constant 8.9875517873682E9 N·m²/coul²</td>
</tr>
<tr>
<td>_g</td>
<td>acceleration of gravity 9.80665 m/s²</td>
</tr>
<tr>
<td>_Gc</td>
<td>gravitational constant 6.6742E-11 m³/kg/s²</td>
</tr>
<tr>
<td>_h</td>
<td>Planck’s constant 6.6260693E-34 J·s</td>
</tr>
<tr>
<td>_k</td>
<td>Boltzmann’s constant 1.3806505E-23 J/K</td>
</tr>
<tr>
<td>_Me</td>
<td>electron rest mass 9.1093826E-31 kg</td>
</tr>
<tr>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>_Mn</td>
<td>neutron rest mass 1.67492728E-27_kg</td>
</tr>
<tr>
<td>_Mp</td>
<td>proton rest mass 1.67262171E-27_kg</td>
</tr>
<tr>
<td>_Na</td>
<td>Avogadro’s number 6.0221415E23 /_mol</td>
</tr>
<tr>
<td>_q</td>
<td>electron charge 1.60217653E-19_coul</td>
</tr>
<tr>
<td>_Rb</td>
<td>Bohr radius 5.291772108E-11_m</td>
</tr>
<tr>
<td>_Rc</td>
<td>molar gas constant 8.314472_J/_mol/°K</td>
</tr>
<tr>
<td>_Rdb</td>
<td>Rydberg constant 10973731.568525 /_m</td>
</tr>
<tr>
<td>_Vm</td>
<td>molar volume 2.2413996E-2_m³/_mol</td>
</tr>
<tr>
<td>_ε0</td>
<td>permittivity of a vacuum 8.8541878176204E-12_F/_m</td>
</tr>
<tr>
<td>_σ</td>
<td>Stefan-Boltzmann constant 5.670400E-8_W/_m²/°K⁴</td>
</tr>
<tr>
<td>_ϕ0</td>
<td>magnetic flux quantum 2.06783372E-15_Wb</td>
</tr>
<tr>
<td>_μ0</td>
<td>permeability of a vacuum 1.2566370614359E-6_N/_A²</td>
</tr>
<tr>
<td>_μb</td>
<td>Bohr magneton 9.27400949E-24_J •_m²/_Wb</td>
</tr>
</tbody>
</table>

Notes:

- The calculator simplifies unit expressions and displays results according to your default units. Therefore, constant values displayed on your screen may appear different from the values in this table.

- For Greek characters, refer to Quick Reference Key Table.
• These values represent the most up-to-date constants available at time of printing from the CODATA Internationally recommended values of the Fundamental Physical Constants available on the National Institute of Standards and Technology (NIST) web site. (http://physics.nist.gov/cuu/Constants/index.html).

### Length

<table>
<thead>
<tr>
<th>_Ang</th>
<th>angstrom</th>
<th>_mi</th>
<th>mile</th>
</tr>
</thead>
<tbody>
<tr>
<td>_au</td>
<td>astronomical unit</td>
<td>_mil</td>
<td>1/1000 inch</td>
</tr>
<tr>
<td>_cm</td>
<td>centimeter</td>
<td>_mm</td>
<td>millimeter</td>
</tr>
<tr>
<td>_fath</td>
<td>fathom</td>
<td>_Nmi</td>
<td>nautical mile</td>
</tr>
<tr>
<td>_fm</td>
<td>fermi</td>
<td>_pc</td>
<td>parsec</td>
</tr>
<tr>
<td>_ft</td>
<td>foot (ENG/US)</td>
<td>_rod</td>
<td>rod</td>
</tr>
<tr>
<td>_in</td>
<td>inch</td>
<td>_yd</td>
<td>yard</td>
</tr>
<tr>
<td>_km</td>
<td>kilometer</td>
<td>_μ</td>
<td>micron</td>
</tr>
<tr>
<td>_ltyr</td>
<td>light year</td>
<td>_Å</td>
<td>angstrom</td>
</tr>
<tr>
<td>_m</td>
<td>meter (SI)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Area

<table>
<thead>
<tr>
<th>_acre</th>
<th>acre</th>
<th>NONE (SI) (ENG/US)</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ha</td>
<td>hectare</td>
<td></td>
</tr>
</tbody>
</table>
### Volume

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unit</th>
<th>Symbol</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>_cup</td>
<td>cup</td>
<td>_ml</td>
<td>milliliter</td>
</tr>
<tr>
<td>_floz</td>
<td>fluid ounce</td>
<td>_pt</td>
<td>pint</td>
</tr>
<tr>
<td>_flozUK</td>
<td>British fluid ounce</td>
<td>_qt</td>
<td>quart</td>
</tr>
<tr>
<td>_gal</td>
<td>gallon</td>
<td>_tbsp</td>
<td>tablespoon</td>
</tr>
<tr>
<td>_galUK</td>
<td>British gallon</td>
<td>_tsp</td>
<td>teaspoon</td>
</tr>
<tr>
<td>_l</td>
<td>liter</td>
<td>NONE</td>
<td>(SI) (ENG/US)</td>
</tr>
</tbody>
</table>

### Time

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unit</th>
<th>Symbol</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>_day</td>
<td>day</td>
<td>_s</td>
<td>second (SI) (ENG/US)</td>
</tr>
<tr>
<td>_hr</td>
<td>hour</td>
<td>_week</td>
<td>week</td>
</tr>
<tr>
<td>_min</td>
<td>minute</td>
<td>_yr</td>
<td>year</td>
</tr>
<tr>
<td>_ms</td>
<td>millisecond</td>
<td>_μs</td>
<td>microsecond</td>
</tr>
<tr>
<td>_ns</td>
<td>nanosecond</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Velocity

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unit</th>
<th>Symbol</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>_knot</td>
<td>knot</td>
<td>_mph</td>
<td>miles per hour</td>
</tr>
<tr>
<td>_kph</td>
<td>kilometers per hour</td>
<td>NONE</td>
<td>(SI) (ENG/US)</td>
</tr>
</tbody>
</table>
## Acceleration

no pre-defined units

## Temperature

<table>
<thead>
<tr>
<th>°C</th>
<th>°Celsius (For °, press [2nd][°].)</th>
</tr>
</thead>
<tbody>
<tr>
<td>°K</td>
<td>°Kelvin</td>
</tr>
<tr>
<td>°F</td>
<td>°Fahrenheit</td>
</tr>
<tr>
<td>°R</td>
<td>°Rankine (no default)</td>
</tr>
</tbody>
</table>

## Luminous Intensity

| _cd | candela (no default) |

## Amount of Substance

| _mol | mole (no default) |

## Mass

<table>
<thead>
<tr>
<th>_amu</th>
<th>atomic mass unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>_oz</td>
<td>ounce</td>
</tr>
<tr>
<td>_gm</td>
<td>gram</td>
</tr>
<tr>
<td>_slug</td>
<td>slug</td>
</tr>
<tr>
<td>_kg</td>
<td>kilogram (SI)</td>
</tr>
<tr>
<td>_ton</td>
<td>ton</td>
</tr>
<tr>
<td>_lb</td>
<td>pound (ENG/US)</td>
</tr>
<tr>
<td>_tonne</td>
<td>metric ton</td>
</tr>
</tbody>
</table>
### Constants and Measurement Units

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>SI Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>mg</td>
<td>milligram</td>
<td>_tonUK long ton</td>
</tr>
<tr>
<td>mton</td>
<td>metric ton</td>
<td></td>
</tr>
</tbody>
</table>

### Force

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>SI Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>dyne</td>
<td>dyne</td>
<td>_N newton (SI)</td>
</tr>
<tr>
<td>kgf</td>
<td>kilogram force</td>
<td>_tonf ton force</td>
</tr>
<tr>
<td>lbf</td>
<td>pound force (ENG/US)</td>
<td></td>
</tr>
</tbody>
</table>

### Energy

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>SI Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Btu</td>
<td>British thermal unit (ENG/US)</td>
<td>_J joule (SI)</td>
</tr>
<tr>
<td>cal</td>
<td>calorie</td>
<td>_kcal kilocalorie</td>
</tr>
<tr>
<td>erg</td>
<td>erg</td>
<td>_kWh kilowatt-hour</td>
</tr>
<tr>
<td>eV</td>
<td>electron volt</td>
<td>_latm liter-atmosphere</td>
</tr>
<tr>
<td>ftlb</td>
<td>foot-pound</td>
<td></td>
</tr>
</tbody>
</table>

### Power

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>SI Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>hp</td>
<td>horsepower (ENG/US)</td>
<td>_W watt (SI)</td>
</tr>
<tr>
<td>kW</td>
<td>kilowatt</td>
<td></td>
</tr>
</tbody>
</table>
### Pressure

<table>
<thead>
<tr>
<th>Unit</th>
<th>Name</th>
<th>Symbol</th>
<th>Unit/Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>atm</td>
<td>atmosphere</td>
<td>atm</td>
<td>pressure</td>
</tr>
<tr>
<td>bar</td>
<td>bar</td>
<td>bar</td>
<td>pressure</td>
</tr>
<tr>
<td>_mmHg</td>
<td>millimeters of mercury</td>
<td>Pa</td>
<td>pascal (SI)</td>
</tr>
<tr>
<td>_inH2O</td>
<td>inches of water</td>
<td>psi</td>
<td>pounds per square inch (ENG/US)</td>
</tr>
<tr>
<td>_inHg</td>
<td>inches of mercury</td>
<td>torr</td>
<td>millimeters of mercury</td>
</tr>
<tr>
<td>_mmH2O</td>
<td>millimeters of water</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Viscosity, Kinematic

<table>
<thead>
<tr>
<th>Unit</th>
<th>Name</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>St</td>
<td>stokes</td>
<td>St</td>
</tr>
</tbody>
</table>

### Viscosity, Dynamic

<table>
<thead>
<tr>
<th>Unit</th>
<th>Name</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>poise</td>
<td>P</td>
</tr>
</tbody>
</table>

### Frequency

<table>
<thead>
<tr>
<th>Unit</th>
<th>Name</th>
<th>Symbol</th>
<th>Unit/Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHz</td>
<td>gigahertz</td>
<td>GHz</td>
<td>frequency</td>
</tr>
<tr>
<td>kHz</td>
<td>kilohertz</td>
<td>kHz</td>
<td>frequency</td>
</tr>
<tr>
<td>Hz</td>
<td>hertz (SI) (ENG/US)</td>
<td>MHz</td>
<td>megahertz</td>
</tr>
</tbody>
</table>
**Electric Current**

<table>
<thead>
<tr>
<th></th>
<th>Unit</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ampere (SI) (ENG/US)</td>
<td>_A</td>
</tr>
<tr>
<td>mA</td>
<td>milliampere</td>
<td>_mA</td>
</tr>
<tr>
<td>kA</td>
<td>kiloampere</td>
<td>_kA</td>
</tr>
<tr>
<td>µA</td>
<td>microampere</td>
<td>_µA</td>
</tr>
</tbody>
</table>

**Charge**

<table>
<thead>
<tr>
<th></th>
<th>Unit</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>coul</td>
<td>coulomb (SI) (ENG/US)</td>
<td>_coul</td>
</tr>
</tbody>
</table>

**Potential**

<table>
<thead>
<tr>
<th></th>
<th>Unit</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>kV</td>
<td>kilovolt</td>
<td>_kV</td>
</tr>
<tr>
<td>mV</td>
<td>millivolt</td>
<td>_mV</td>
</tr>
<tr>
<td>V</td>
<td>volt (SI) (ENG/US)</td>
<td>_V</td>
</tr>
<tr>
<td>volt</td>
<td>volt (SI) (ENG/US)</td>
<td>_volt</td>
</tr>
</tbody>
</table>

**Resistance**

<table>
<thead>
<tr>
<th></th>
<th>Unit</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>kΩ</td>
<td>kilo ohm</td>
<td>_kΩ</td>
</tr>
<tr>
<td>Ω</td>
<td>ohm (SI) (ENG/US)</td>
<td>_Ω</td>
</tr>
<tr>
<td>MΩ</td>
<td>megaohm</td>
<td>_MΩ</td>
</tr>
</tbody>
</table>

**Conductance**

<table>
<thead>
<tr>
<th></th>
<th>Unit</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>mho</td>
<td>mho (ENG/US)</td>
<td>_mho</td>
</tr>
<tr>
<td>µmho</td>
<td>micromho</td>
<td>_µmho</td>
</tr>
</tbody>
</table>
## Capacitance

<table>
<thead>
<tr>
<th>_F</th>
<th>farad (SI) (ENG/US)</th>
<th>_pF</th>
<th>picofarad</th>
</tr>
</thead>
<tbody>
<tr>
<td>_nF</td>
<td>nanofarad</td>
<td>_µF</td>
<td>microfarad</td>
</tr>
</tbody>
</table>

## Mag Field Strength

<table>
<thead>
<tr>
<th>_Oe</th>
<th>oersted</th>
<th>NONE (SI) (ENG/US)</th>
</tr>
</thead>
</table>

## Mag Flux Density

<table>
<thead>
<tr>
<th>_Gs</th>
<th>gauss</th>
<th>_T</th>
<th>tesla (SI) (ENG/US)</th>
</tr>
</thead>
</table>

## Magnetic Flux

<table>
<thead>
<tr>
<th>_Wb</th>
<th>weber (SI) (ENG/US)</th>
</tr>
</thead>
</table>

## Inductance

<table>
<thead>
<tr>
<th>_henry</th>
<th>henry (SI) (ENG/US)</th>
<th>_nH</th>
<th>nanohenry</th>
</tr>
</thead>
<tbody>
<tr>
<td>_mH</td>
<td>millihenry</td>
<td>_µH</td>
<td>microhenry</td>
</tr>
</tbody>
</table>
To graph one or more $y(x)$ functions, use the general steps shown below. For a detailed description of each step, refer to the following pages. You may not need to do all the steps each time you graph a function.

**Graphing Functions**

1. Set **Graph** mode (MODE) to **FUNCTION**. Also set **Angle** mode, if necessary.

2. Define x and y components on Y= Editor (Y [Y=]).

3. Select (F4) which defined functions to graph.

**Note:** To turn off any stat data plots, press F5 5 or use F4 to deselect them.
4. Set the display style for a function. 

\[ \text{2nd } [F6] \]

This is optional. For multiple equations, this helps visually distinguish one from another.

5. Define the viewing window \( \text{[WINDOW]} \).

\[ F2 \text{ Zoom} \] also changes the viewing window.

6. Change the graph format if necessary.

\[ F1 \ 9 \]

– or –

\[ \text{[GRAPH]} \]

7. Graph the selected functions (\( \text{[GRAPH]} \)).

**Exploring the Graph**

From the Graph screen, you can:

- Display the coordinates of any pixel by using the free-moving cursor, or of a plotted point by tracing a function.
- Use the \( F2 \text{ Zoom} \) toolbar menu to zoom in or out on a portion of the graph.
• Use the F5 Math toolbar menu to find a zero, minimum, maximum, etc.

**Setting the Graph Mode**

Before graphing \( y(x) \) functions, you must select FUNCTION graphing. You may also need to set the Angle mode, which affects how the TI-89 Titanium graphs trigonometric functions.

**Graph Mode**

1. Press MODE to display the MODE dialog box, which shows the current mode settings.

2. Set the Graph mode to **FUNCTION**. Refer to “Setting Modes” in *Operating the Calculator*.

   For graphs that do not use complex numbers, set **Complex Format = REAL**.

   Otherwise, it may affect graphs that use powers, such as \( x^{1/3} \).

While this module specifically describes \( y(x) \) function graphs, the calculator lets you select from six Graph mode settings.

<table>
<thead>
<tr>
<th>Graph Mode Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTION</td>
<td>( y(x) ) functions</td>
</tr>
<tr>
<td>PARAMETRIC</td>
<td>( x(t) ) and ( y(t) ) parametric equations</td>
</tr>
</tbody>
</table>
Angle Mode

When using trigonometric functions, set the Angle mode for the units (Radian, Degree or Gradian) in which you want to enter and display angle values.

Checking the Status Line

To see the current Graph mode and Angle mode, check the status line at the bottom of the screen.

<table>
<thead>
<tr>
<th>MAIN</th>
<th>RAD AUTO</th>
<th>FUNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle Mode</td>
<td>Graph Mode</td>
<td></td>
</tr>
</tbody>
</table>

Defining Functions for Graphing

In FUNCTION graphing mode, you can graph functions named $y_1(x)$ through $y_{99}(x)$. To define and edit these functions, use the Y= Editor. (The Y= Editor lists function names for
the current graphing mode. For example, in POLAR graphing mode, function names are \( r1(\theta), r2(\theta), \text{etc.} \)

**Defining a New Function**

1. Press \( \blacklozenge \) \( Y= \) to display the Y= Editor.

2. Press \( \blacklozenge \) and \( \blacklozenge \) to move the cursor to any undefined function. (Use \( \text{2nd} \) \( \blacklozenge \) and \( \text{2nd} \) \( \blacklozenge \) to scroll one page at a time.)

3. Press \( \text{ENTER} \) or \( \text{F3} \) to move the cursor to the entry line.

4. Type the expression to define the function.
   - The independent variable in function graphing is \( x \).
   - The expression can refer to other variables, including matrices, lists, and other functions. Only floats and lists of floats will produce a plot.

**Note:** For an undefined function, you do not need to press \( \text{ENTER} \) or \( \text{F3} \). When you begin typing, the cursor moves to the entry line.

5. When you complete the expression, press \( \text{ENTER} \).
The function list now shows the new function, which is automatically selected for graphing.

**Note:** If you accidentally move the cursor to the entry line, press **ESC** to move it back to the function list.

**Editing a Function**

From the Y= Editor:

1. Press **C** and **C** to highlight the function.
2. Press **ENTER** or **F3** to move the cursor to the entry line.
3. Do any of the following:
   - Use **D** and **D** to move the cursor within the expression and edit it. Refer to “Editing an Expression in the Entry Line” in *Operating the Calculator*.
   - or –
   - Press **CLEAR** once or twice to clear the old expression, and then type the new one.
4. Press **ENTER**.

The function list now shows the edited function, which is automatically selected for graphing.

**Note:** To cancel any editing changes, press **ESC** instead of **ENTER**.
Clearing a Function

From the Y= Editor:

<table>
<thead>
<tr>
<th>To erase:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A function from the function list</td>
<td>Highlight the function and press ← or [CLEAR].</td>
</tr>
<tr>
<td>A function from the entry line</td>
<td>Press [CLEAR] once or twice (depending on the cursor’s location) and then press [ENTER].</td>
</tr>
<tr>
<td>All functions</td>
<td>Press [F1] and then select 8:Clear Functions. When prompted for confirmation, press [ENTER].</td>
</tr>
</tbody>
</table>

**Note:** \[F1\] 8 does not erase any stat plots.

You don’t have to clear a function to prevent it from being graphed. You can select the functions you want to graph.

**Shortcuts to Move the Cursor**

From the Y= Editor:

<table>
<thead>
<tr>
<th>Press:</th>
<th>To:</th>
</tr>
</thead>
<tbody>
<tr>
<td>[†] [↑] or [†] [↓]</td>
<td>Go to function 1 or to the last defined function, respectively. If the cursor is on or past the last defined function, [†] [↓] goes to function 99.</td>
</tr>
</tbody>
</table>
From the Home Screen or a Program

You can also define and evaluate a function from the Home screen or a program.

- Use the Define and Graph commands. Refer to:
  - “Graphing a Function Defined on the Home Screen” and “Graphing a Piecewise Defined Function” in Additional Graphing Topics.
  - “Overview of Entering a Function” in Programming.
- Store an expression directly to a function variable. Refer to:
  - “Storing and Recalling Variable Values” in Operating the Calculator.

Note: User-defined functions can have almost any name. However, if you want them to appear in the Y= Editor, use function names y1(x), y2(x), etc.

Selecting Functions to Graph

Regardless of how many functions are defined in the Y= Editor, you can select the ones you want to graph.

Selecting or Deselecting Functions

Press [Y=] to display the Y= Editor.

A “✓” indicates which functions will be graphed the next time you display the Graph screen.
You don’t have to select a function when you enter or edit it; it is selected automatically. To turn off any stat plots, press **F5** 5 or use **F4** to deselect them.

**From the Home Screen or a Program**

You can also select or deselect functions from the Home screen or a program.
• Use the **FnOn** and **FnOff** commands (available from the Home screen’s **F4 Other** toolbar menu) for functions. Refer to the *Technical Reference* module.

• Use the **PlotsOn** and **PlotsOff** commands for stat plots. Refer to the *Technical Reference* module.

### Setting the Display Style for a Function

For each defined function, you can set a style that specifies how that function will be graphed. This is useful when graphing multiple functions. For example, set one as a solid line, another as a dotted line, etc.

### Displaying or Changing a Function’s Style

From the Y= Editor:

1. Move the cursor to highlight the applicable function.
2. Select the **Style** menu and press: **[2nd] [F6]**

- Although the Line item is initially highlighted, the function’s current style is indicated by a ✔ mark.
- To exit the menu without making a change, press **[ESC]**.
3. To make a change, select the applicable style.

<table>
<thead>
<tr>
<th>Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>Connects plotted points with a line. This is the default.</td>
</tr>
<tr>
<td>Dot</td>
<td>Displays a dot at each plotted point.</td>
</tr>
<tr>
<td>Square</td>
<td>Displays a solid box at each plotted point.</td>
</tr>
<tr>
<td>Thick</td>
<td>Connects plotted points with a thick line.</td>
</tr>
<tr>
<td>Animate</td>
<td>A round cursor moves along the leading edge of the graph but does not leave a path.</td>
</tr>
<tr>
<td>Path</td>
<td>A round cursor moves along the leading edge of the graph and does leave a path.</td>
</tr>
<tr>
<td>Above</td>
<td>Shades the area above the graph.</td>
</tr>
<tr>
<td>Below</td>
<td>Shades the area below the graph.</td>
</tr>
</tbody>
</table>

To set Line as the style for all functions, press \( \text{F5} \) and select 4:Reset Styles.

If You Use Above or Below Shading

The TI-89 Titanium has four shading patterns, used on a rotating basis. If you set one function as shaded, it uses the first pattern. The next shaded function uses the second pattern, etc. The fifth shaded function reuses the first pattern.

When shaded areas intersect, their patterns overlap.
From the Home Screen or a Program

You can also set a function’s style from the Home screen or a program. Refer to the Style command in the Technical Reference module.

Defining the Viewing Window

The viewing window represents the portion of the coordinate plane displayed on the Graph screen. By setting Window variables, you can define the viewing window’s boundaries and other attributes. Function graphs, parametric graphs, etc., have their own independent set of Window variables.

Displaying Window Variables in the Window Editor

Press \[ \text{WINDOW} \] to display the Window Editor.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xmin, xmax, ymin, ymax</td>
<td>Boundaries of the viewing window.</td>
</tr>
</tbody>
</table>
To turn off tick marks, set `xscl=0` and/or `yscl=0`. Small values of `xres` improve the graph’s resolution but may reduce the graphing speed.

### Changing the Values

From the Window Editor:

1. Move the cursor to highlight the value you want to change.
2. Do any of the following:
   - Type a value or an expression. The old value is erased when you begin typing.
     – or –
   - Press `CLEAR` to clear the old value; then type the new one.
     – or –
   - Press ◁ or ◀ to remove the highlighting; then edit the value.

Values are stored as you type them; you do not need to press [ENTER]. [ENTER] simply moves the cursor to the next Window variable. If you type an expression, it is evaluated when you move the cursor to a different Window variable or leave the Window Editor.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>xscl, yscl</code></td>
<td>Distance between tick marks on the x and y axes.</td>
</tr>
<tr>
<td><code>xres</code></td>
<td>Sets pixel resolution (1 through 10) for function graphs. The default is 2.</td>
</tr>
</tbody>
</table>
  - At 1, functions are evaluated and graphed at each pixel along the x axis.
  - At 10, functions are evaluated and graphed at every 10th pixel along the x axis. |
From the Home Screen or a Program

You can also store values directly to the Window variables from the Home screen or a program. Refer to “Storing and Recalling Variable Values” in *Operating the Calculator*.

Changing the Graph Format

You can set the graph format to show or hide reference elements such as the axes, a grid, and the cursor’s coordinates. Function graphs, parametric graphs, etc., have their own independent set of graph formats.

Displaying Graph Format Settings

From the Y= Editor, Window Editor, or Graph screen, press \[F1\] and select 9:Format.

- The GRAPH FORMATS dialog box shows the current settings.
- To exit without making a change, press \[ESC\].

You also can display the GRAPH FORMATS dialog box from the Y= Editor, Window Editor, or Graph screen. Press: \[DISP\] \[\[\]

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinates</td>
<td>Shows cursor coordinates in rectangular (RECT) or polar (POLAR) form, or hides (OFF) the coordinates.</td>
</tr>
</tbody>
</table>
To turn off tick marks, define the viewing window so that $x_{scl}$ and/or $y_{scl} = 0$.

### Changing Settings

From the GRAPH FORMATS dialog box:

1. Move the cursor to highlight the format setting.
2. Press $\text{ENT}$ to display a menu of valid settings for that format.
3. Select a setting. Either:
   - Move the cursor to highlight the setting, and then press $\text{ENT}$.
   - or –
   - Press the number for that setting.
4. After changing all applicable format settings, press \texttt{ENTER} to save your changes and close the \texttt{GRAPH FORMATS} dialog box.

\textbf{Note:} To cancel a menu or exit the dialog box without saving any changes, use \texttt{ESC} instead of \texttt{ENTER}.

**Graphing the Selected Functions**

When you are ready to graph the selected functions, display the Graph screen. This screen uses the display style and viewing window that you previously defined.

**Displaying the Graph Screen**

Press \texttt{[GRAPH]}. The TI-89 Titanium automatically graphs the selected functions.

![Graph Screen](image)

\texttt{BUSY} indicator shows while graphing is in progress.

If you select an \texttt{F2 Zoom} operation from the \texttt{Y= Editor} or \texttt{Window Editor}, the TI-89 Titanium automatically displays the Graph screen.

**Interrupting Graphing**

While graphing is in progress:
• To pause graphing temporarily, press \textbf{[ENTER]}. (The \textbf{PAUSE} indicator replaces \textbf{BUSY}.) To resume, press \textbf{[ENTER]} again.

• To cancel graphing, press \textbf{[ON]}. To start graphing again from the beginning, press \textbf{[F4]} (ReGraph).

\textbf{If You Need to Change the Viewing Window}

Depending on various settings, a function may be graphed such that it is too small, too large, or offset too far to one side of the screen. To correct this:

• Redefine the viewing window with different boundaries.
• Use a Zoom operation.

\textbf{Smart Graph}

When you display the Graph screen, the Smart Graph feature displays the previous window contents immediately, provided nothing has changed that requires regraphing.

Smart Graph updates the window and regraphs only if you have:

• Changed a mode setting that affects graphing, a function’s graphing attribute, a Window variable, or a graph format.
• Selected or deselected a function or stat plot. (If you only select a new function, Smart Graph adds that function to the Graph screen.)
• Changed the definition of a selected function or the value of a variable in a selected function.
• Cleared a drawn object.
• Changed a stat plot definition.

**Displaying Coordinates with the Free-Moving Cursor**

To display the coordinates of any location on the Graph screen, use the free-moving cursor. You can move the cursor to any pixel on the screen; the cursor is not confined to a graphed function.

**Free-Moving Cursor**

When you first display the Graph screen, no cursor is visible. To display the cursor, press a cursor pad arrow. The cursor moves from the center of the screen, and its coordinates are displayed.

\[ y_1(x) = x^2 \]

The “c” indicates these are cursor coordinates. The values are stored in the \( xc \) and \( yc \) system variables. Rectangular coordinates use \( xc \) and \( yc \). Polar coordinates use \( rc \) and \( \theta c \).

If your screen does not show coordinates, set the graph format so that Coordinates = **RECT** or **POLAR**. Press:

\[ \Box \] \[ \Box \]

---

**To move the free-moving cursor:**

<table>
<thead>
<tr>
<th>Press:</th>
</tr>
</thead>
<tbody>
<tr>
<td>To an adjoining pixel</td>
</tr>
<tr>
<td>A cursor pad arrow for any direction.</td>
</tr>
</tbody>
</table>
### To move the free-moving cursor:

<table>
<thead>
<tr>
<th>Press:</th>
<th>In increments of 10 pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd</td>
<td>2nd and then a cursor pad arrow.</td>
</tr>
</tbody>
</table>

### Note:
To hide the cursor and its coordinates temporarily, press **CLEAR**, **ESC**, or **ENTER**. The next time you move the cursor, it moves from its last position.

When you move the cursor to a pixel that appears to be “on” the function, it may be near the function but not on it.

To increase the accuracy:

- Use the **Trace** tool described on the next page to display coordinates that are on the function.
- Use a **Zoom** operation to zoom in on a portion of the graph.
Tracing a Function

To display the exact coordinates of any plotted point on a graphed function, use the \( \text{F3} \) Trace tool. Unlike the free-moving cursor, the trace cursor moves only along a function’s plotted points.

Beginning a Trace

From the Graph screen, press \( \text{F3} \).

The trace cursor appears on the function, at the middle x value on the screen. The cursor’s coordinates are displayed at the bottom of the screen.

If multiple functions are graphed, the trace cursor appears on the lowest-numbered function selected in the Y= Editor. The function number is shown in the upper right part of the screen.

If any stat plots are graphed, the trace cursor appears on the lowest-numbered stat plot.

Moving along a Function

<table>
<thead>
<tr>
<th>To move the trace cursor:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>To the previous or next plotted point</td>
<td>Press ( \text{left} ) or ( \text{right} ).</td>
</tr>
<tr>
<td>Approximately 5 plotted points (it may be more or less than 5, depending on the \textbf{xres} Window variable)</td>
<td>Press ( \text{2nd} ) ( \text{left} ) or ( \text{2nd} ) ( \text{right} ).</td>
</tr>
</tbody>
</table>
To move the trace cursor:

<table>
<thead>
<tr>
<th>To do</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>To a specified x value on the function</td>
<td>Type the x value and press &lt;code&gt;ENTER&lt;/code&gt;.</td>
</tr>
</tbody>
</table>

**Note:** If you enter an x value, it must be between <code>xmin</code> and <code>xmax</code>.

The trace cursor moves only from plotted point to plotted point along the function, not from pixel to pixel.

If your screen does not show coordinates, set the graph format so that <code>Coordinates = RECT</code> or <code>POLAR</code>. Press:

<code>▼</code> <code>►</code>

Each displayed y value is calculated from the x value; that is, <code>y=yn(x)</code>. If the function is undefined at an x value, the y value is blank.

You can continue to trace a function that goes above or below the viewing window. You cannot see the cursor as it moves in that “off the screen” area, but the displayed coordinate values show its correct coordinates.

**Note:** Use QuickCenter to trace a function that goes above or below the window.
Moving from Function to Function

Press ↳ or ↱ to move to the previous or next selected function at the same x value. The new function number is shown on the screen.

The “previous or next” function is based on the order of the selected functions in the Y= Editor, not the appearance of the functions as graphed on the screen.

Automatic Panning

If you trace a function off the left or right edge of the screen, the viewing window automatically pans to the left or right. There is a slight pause while the new portion of the graph is drawn.

Before automatic pan

After automatic pan

After an automatic pan, the cursor continues tracing.

Note: Automatic panning does not work if stat plots are displayed or if a function uses a shaded display style.
Using QuickCenter

If you trace a function off the top or bottom of the viewing window, you can press **ENTER** to center the viewing window on the cursor location.

![Graph before QuickCenter](image1)

![Graph after QuickCenter](image2)

Before using QuickCenter  After using QuickCenter

After QuickCenter, the cursor stops tracing. If you want to continue tracing, press **F3**.

You can use QuickCenter at any time during a trace, even when the cursor is still on the screen.

Canceling Trace

To cancel a trace at any time, press **ESC**.

A trace is also canceled when you display another application screen such as the Y= Editor. When you return to the Graph screen and press **F3** to begin tracing:

- If Smart Graph regraphed the screen, the cursor appears at the middle x value.
- If Smart Graph does not regraph the screen, the cursor appears at its previous location (before you displayed the other application).
Using Zooms to Explore a Graph

The \texttt{F2 Zoom} toolbar menu has several tools that let you adjust the viewing window. You can also save a viewing window for later use.

\section*{Overview of the Zoom Menu}

Press \texttt{F2} from the Y= Editor, Window Editor, or Graph screen.

Procedures for using \texttt{ZoomBox}, \texttt{ZoomIn}, \texttt{ZoomOut}, \texttt{ZoomStd}, \texttt{Memory}, and \texttt{SetFactors} are given later in this section.

For more information about the other items, refer to the \textit{Technical Reference} module.

\textbf{Note:} If you select a \texttt{Zoom} tool from the Y=Editor or Window Editor, the TI-89 Titanium automatically displays the Graph screen.

<table>
<thead>
<tr>
<th>Zoom Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{ZoomBox}</td>
<td>Lets you draw a box and zoom in on that box.</td>
</tr>
<tr>
<td>\texttt{ZoomIn, ZoomOut}</td>
<td>Lets you select a point and zoom in or out by an amount defined by \texttt{SetFactors}.</td>
</tr>
<tr>
<td>\texttt{ZoomDec}</td>
<td>Sets $\Delta x$ and $\Delta y$ to .1, and centers the origin.</td>
</tr>
<tr>
<td>\texttt{ZoomSqr}</td>
<td>Adjusts Window variables so that a square or circle is shown in correct proportion (instead of a rectangle or ellipse).</td>
</tr>
</tbody>
</table>
### Zoom Tool Description

<table>
<thead>
<tr>
<th>Zoom Tool</th>
<th>Description</th>
</tr>
</thead>
</table>
| ZoomStd   | Sets Window variables to their default values. $\begin{align*} &\text{xmin} = -10 \quad \text{ymin} = -10 \quad \text{xres} = 2 \\
&\text{xmax} = 10 \quad \text{ymax} = 10 \\
&\text{xscl} = 1 \quad \text{yscl} = 1 \end{align*}$ |
| ZoomTrig  | Sets Window variables to preset values that are often appropriate for graphing trig functions. Centers the origin and sets: $\begin{align*} &\Delta x = \pi/24 \quad (\text{.130899... radians or 7.5 degrees}) \quad \text{ymin} = -4 \\
&\Delta x = \pi/2 \quad (1.570796... \text{ radians or 90 degrees}) \quad \text{ymax} = 4 \\
&\text{xscl} = \pi/2 \quad \text{yscl} = 0.5 \end{align*}$ |
| ZoomInt   | Lets you select a new center point, and then sets $\Delta x$ and $\Delta y$ to 1 and sets $\text{xscl}$ and $\text{yscl}$ to 10. |
| ZoomData  | Adjusts Window variables so that all selected stat plots are in view. |
| ZoomFit   | Adjusts the viewing window to display the full range of dependent variable values for the selected functions. In function graphing, this maintains the current $\text{xmin}$ and $\text{xmax}$ and adjusts $\text{ymin}$ and $\text{ymax}$. |
| Memory    | Lets you store and recall Window variable settings so that you can recreate a custom viewing window. |
| SetFactors| Lets you set Zoom factors for ZoomIn and ZoomOut. |

$\Delta x$ and $\Delta y$ are the distances from the center of one pixel to the center of an adjoining pixel.
Zooming In with a Zoom Box

1. From the F2 Zoom menu, select 1:ZoomBox.
   The screen prompts for 1st Corner?

2. Move the cursor to any corner of the box you want to define, and then press ENTER.
   The cursor changes to a small square, and the screen prompts for 2nd Corner?
   **Note:** To move the cursor in larger increments, use 2nd ▶, 2nd ◀, etc.

3. Move the cursor to the opposite corner of the zoom box.
   As you move the cursor, the box stretches.

4. When you have outlined the area you want to zoom in on, press ENTER.
   The Graph screen shows the zoomed area. You can cancel ZoomBox by pressing ESC before you press ENTER.
Zooming In and Out on a Point

1. From the [F2] Zoom menu, select 2:ZoomIn or 3:ZoomOut.

   A cursor appears, and the screen prompts for New Center?

2. Move the cursor to the point where you want to zoom in or out, and then press [ENTER].

   The TI-89 Titanium adjusts the Window variables by the Zoom factors defined in SetFactors.

   - For a ZoomIn, the x variables are divided by xFact, and the y variables are divided by yFact.
     
     new xmin = xmin/xFact , etc.

   - For a ZoomOut, the x variables are multiplied by xFact, and the y variables are multiplied by yFact.
     
     new xmin = xmin * xFact , etc.

Changing Zoom Factors

The Zoom factors define the magnification and reduction used by ZoomIn and ZoomOut.
1. From the **F2** **Zoom** menu, select **C:SetFactors** to display the **ZOOM FACTORS** dialog box.

    ![ZOOM FACTORS dialog box]

    Zoom factors must be $\geq 1$, but they do not have to be integers. The default setting is 4.

    **Note:** To exit without saving any changes, press **ESC**.

2. Use $\bigcirc$ and $\bigtriangledown$ to highlight the value you want to change. Then:
   - Type the new value. The old value is cleared automatically when you begin typing.
   - or –
   - Press $\bigcirc$ or $\bigtriangledown$ to remove the highlighting, and then edit the old value.

3. Press **ENTER** (after typing in an input box, you must press **ENTER** twice) to save any changes and exit the dialog box.

**Saving or Recalling a Viewing Window**

After using various **Zoom** tools, you may want to return to a previous viewing window or save the current one.

1. From the **F2** **Zoom** menu, select **B:Memory** to display its submenu.

    ![Zoom menu]

    1:ZoomPrev
    2:zoomSto
    3:zoomRcl

2. Select the applicable item.
<table>
<thead>
<tr>
<th>Select:</th>
<th>To:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:ZoomPrev</td>
<td>Return to the viewing window displayed before the previous zoom.</td>
</tr>
<tr>
<td>2:ZoomSto</td>
<td>Save the current viewing window. (The current Window variable values are stored to the system variables $zxmin$, $zxmax$, etc.)</td>
</tr>
<tr>
<td>3:ZoomRcl</td>
<td>Recall the viewing window last stored with ZoomSto.</td>
</tr>
</tbody>
</table>

**Note:** You can store only one set of Window variable values at a time. Storing a new set overwrites the old set.

**Restoring the Standard Viewing Window**

You can restore the Window variables to their default values at any time. From the F2 Zoom menu, select 6:ZoomStd.

**Using Math Tools to Analyze Functions**

On the Graph screen, the F5 Math toolbar menu has several tools that help you analyze graphed functions.
Overview of the Math Menu

Press \[ F_5 \] from the Graph screen.

On the Derivatives submenu, only \( \frac{dy}{dx} \) is available for function graphing. The other derivatives are available for other graphing modes (parametric, polar, etc.).

<table>
<thead>
<tr>
<th>Math Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Evaluates a selected ( y(x) ) function at a specified ( x ) value.</td>
</tr>
<tr>
<td>Zero, Minimum, Maximum</td>
<td>Finds a zero (x-intercept), minimum, or maximum point within an interval.</td>
</tr>
<tr>
<td>Intersection</td>
<td>Finds the intersection of two functions.</td>
</tr>
<tr>
<td>Derivatives</td>
<td>Finds the derivative (slope) at a point.</td>
</tr>
<tr>
<td>( \int f(x) , dx )</td>
<td>Finds the approximate numerical integral over an interval.</td>
</tr>
<tr>
<td>Inflection</td>
<td>Finds the inflection point of a curve, where its second derivative changes sign (where the curve changes concavity).</td>
</tr>
<tr>
<td>Distance</td>
<td>Draws and measures a line between two points on the same function or on two different functions.</td>
</tr>
<tr>
<td>Tangent</td>
<td>Draws a tangent line at a point and displays its equation.</td>
</tr>
<tr>
<td>Arc</td>
<td>Finds the arc length between two points along a curve.</td>
</tr>
</tbody>
</table>
### Math Tool Description

<table>
<thead>
<tr>
<th>Math Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shade</td>
<td>Depends on the number of functions graphed.</td>
</tr>
<tr>
<td></td>
<td>• If only one function is graphed, this shades the function's area above or below the x axis.</td>
</tr>
<tr>
<td></td>
<td>• If two or more functions are graphed, this shades the area between any two functions within an interval.</td>
</tr>
</tbody>
</table>

**Note:** For Math results, cursor coordinates are stored in system variables xc and yc (rc and θc if you use polar coordinates). Derivatives, integrals, distances, etc., are stored in the system variable `sysMath`.

**Finding y(x) at a Specified Point**

1. From the **Graph** screen, press `F5` and select **1:Value**.

2. Type the x value, which must be a real value between `xmin` and `xmax`. The value can be an expression.

3. Press `ENTER`.

   The cursor moves to that x value on the first function selected in the Y= Editor, and its coordinates are displayed.

4. Press ◻ or ◼ to move the cursor between functions at the entered x value. The corresponding y value is displayed.

   If you press ◆ or ◅, the free-moving cursor appears. You may not be able to move it back to the entered x value.
You can also display function coordinates by tracing the function (F3), typing an x value, and pressing [ENTER].

**Finding a Zero, Minimum, or Maximum within an Interval**

1. From the **Graph** screen, press F5 and select 2:Zero, 3:Minimum, or 4:Maximum.
2. As necessary, use ◀ and ▶ to select the applicable function.
   
   **Note:** Typing x values is a quick way to set bounds.
3. Set the lower bound for x. Either use ◄ and ► to move the cursor to the lower bound or type its x value.
4. Press [ENTER]. A ▶ at the top of the screen marks the lower bound.

5. Set the upper bound, and press [ENTER].
   
   The cursor moves to the solution, and its coordinates are displayed.

**Finding the Intersection of Two Functions within an Interval**

1. From the **Graph** screen, press F5 and select 5:Intersection.
2. Select the first function, using ◀ or ▶ as necessary, and press [ENTER]. The cursor moves to the next graphed function.
3. Select the second function, and press [ENTER].
4. Set the lower bound for x. Either use ◄ and ► to move the cursor to the lower bound or type its x value.
5. Press \[ \text{ENTER} \]. A \( \text{►} \) at the top of the screen marks the lower bound.

6. Set the upper bound, and press \[ \text{ENTER} \].

   The cursor moves to the intersection, and its coordinates are displayed.

\[
y_2(x) = 2x - 7
\]

---

**Finding the Derivative (Slope) at a Point**

1. From the **Graph** screen, press \[ F5 \] and select **6:Derivatives**. Then select **1:dy/dx** from the submenu.
2. As necessary, use \( \odot \) and \( \odot \) to select the applicable function.
3. Set the derivative point. Either move the cursor to the point or type its x value.
4. Press \[ \text{ENTER} \].

   The derivative at that point is displayed.

---

**Finding the Numerical Integral over an Interval**

1. From the **Graph** screen, press \[ F5 \] and select **7:∫f(x)dx**.
2. As necessary, use \( \odot \) and \( \odot \) to select the applicable function.

   **Note:** Typing x values is a quick way to set the limits.
3. Set the lower limit for x. Either use ▼ and ▲ to move the cursor to the lower limit or type its x value.

4. Press ENTER. A ▶ at the top of the screen marks the lower limit.

   **Note:** To erase the shaded area, press F4 (ReGraph).

5. Set the upper limit, and press ENTER.

   The interval is shaded, and its approximate numerical integral is displayed.

---

**Finding an Inflection Point within an Interval**

1. From the Graph screen, press F5 and select 8:Inflection.

2. As necessary, use  and  to select the applicable function.

3. Set the lower bound for x. Either use ▼ and ▲ to move the cursor to the lower bound or type its x value.

4. Press ENTER. A ▶ at the top of the screen marks the lower bound.

5. Set the upper bound, and press ENTER.

   The cursor moves to the inflection point (if any) within the interval, and its coordinates are displayed.

---

**Finding the Distance between Two Points**

1. From the Graph screen, press F5 and select 9:Distance.
2. As necessary, use \( \Phi \) and \( \Theta \) to select the function for the first point.

3. Set the first point. Either use \( \Upsilon \) or \( \Lambda \) to move the cursor to the point or type its \( x \) value.

4. Press \( \text{ENTER} \). A + marks the point.

5. If the second point is on a different function, use \( \Phi \) and \( \Theta \) to select the function.

6. Set the second point. (If you use the cursor to set the point, a line is drawn as you move the cursor.)

7. Press \( \text{ENTER} \).

   The distance between the two points is displayed, along with the connecting line.

**Drawing a Tangent Line**

1. From the **Graph** screen, press \( \text{F5} \) and select **A:Tangent**.

2. As necessary, use \( \Phi \) and \( \Theta \) to select the applicable function.

   **Note:** To erase a drawn tangent line, press \( \text{F4} \) (**ReGraph**).

3. Set the tangent point. Either move the cursor to the point or type its \( x \) value.

4. Press \( \text{ENTER} \).

   The tangent line is drawn, and its equation is displayed.
Finding an Arc Length

1. From the Graph screen, press [F5] and select B:Arc.
2. As necessary, use [◄] and [►] to select the applicable function.
3. Set the first point of the arc. Either use [▼] or [▲] to move the cursor or type the x value.
4. Press [ENTER]. A + marks the first point.
5. Set the second point, and press [ENTER].
   A + marks the second point, and the arc length is displayed.

Shading the Area between a Function and the x Axis

You must have only one function graphed. If you graph two or more functions, the Shade tool shades the area between two functions.

1. From the Graph screen, press [F5] and select C:Shade. The screen prompts for Above X axis?
2. Select one of the following. To shade the function’s area:
   • Above the x axis, press [ENTER].
   • Below the x axis, press: [alpha] N
3. Set the lower bound for x. Either use ④ and ⑥ to move the cursor to the lower bound or type its x value.

   **Note:** If you do not press ④ or ⑥, or type an x value when setting the lower and upper bound, xmin and xmax will be used as the lower and upper bound, respectively.

4. Press ENTER. A ▶ at the top of the screen marks the lower bound.

5. Set the upper bound, and press ENTER. The bounded area is shaded. To erase the shaded area, press F4 (ReGraph).

**Shading the Area between Two Functions within an Interval**

You must have at least two functions graphed. If you graph only one function, the Shade tool shades the area between the function and the x axis.

1. From the Graph screen, press F5 and select C:Shade. The screen prompts for Above?

2. As necessary, use ③ or ② to select a function. (Shading will be above this function.)

3. Press ENTER. The cursor moves to the next graphed function, and the screen prompts for Below?

4. As necessary, use ③ or ② to select another function. (Shading will be below this function.)

5. Press ENTER.
6. Set the lower bound for \( x \). Either use \( \textcircled{<} \) and \( \textcircled{>} \) to move the cursor to the lower bound or type its \( x \) value.

**Note:** If you do not press \( \textcircled{<} \) or \( \textcircled{>} \), or type an \( x \) value when setting the lower and upper bound, \( \text{xmin} \) and \( \text{xmax} \) will be used as the lower and upper bound, respectively.

7. Press [ENTER]. A \( \textgreater \) at the top of the screen marks the lower bound.

8. Set the upper bound, and press [ENTER].
   The bounded area is shaded. To erase the shaded area, press [F4] (ReGraph).
Overview of Steps in Graphing Polar Equations

To graph polar equations, use the same general steps used for \( y(x) \) functions as described in *Basic Function Graphing*. Any differences that apply to polar equations are described on the following pages.

Graphing Polar Equations

1. Set **Graph** mode (MODE) to **POLAR**. Also set **Angle** mode, if necessary.

2. Define \( x \) and \( y \) components on Y= Editor ([Y=]).

3. Select (F4) which defined equations to graph. Select the \( x \) or \( y \) component, or both.

   **Note:** To turn off any stat data plots press [F5] 5 or use [F4] to deselect them.
4. Set the display style for an equation. You can set either the x or y component.

This is optional. For multiple equations, this helps visually distinguish one from another.

5. Define the viewing window ([WINDOW]).

Zoom also changes the viewing window.

6. Change the graph format if necessary.

7. Graph the selected equations ([GRAPH]).

Exploring the Graph

From the Graph screen, you can:

• Display the coordinates of any pixel by using the free-moving cursor, or of a plotted point by tracing a polar equation.
• Use the F2 Zoom toolbar menu to zoom in or out on a portion of the graph.
• Use the F5 Math toolbar menu to find derivatives, tangents, etc. Some menu items are not available for polar graphs.

**Differences in Polar and Function Graphing**

This module assumes that you already know how to graph \( y(x) \) functions as described in *Basic Function Graphing*. This section describes the differences that apply to polar equations.

**Setting the Graph Mode**

Use MODE to set Graph = POLAR before you define equations or set Window variables. The Y= Editor and the Window Editor let you enter information for the current Graph mode setting only.

You should also set the Angle mode to the units (Radian or Degree) you want to use for \( \theta \).

**Defining Polar Equations on the Y= Editor**

You can define polar equations for \( r_1(\theta) \) through \( r_{99}(\theta) \).
You can use the **Define** command from the Home screen (see the *Technical Reference* module) to define functions and equations for any graphing mode, regardless of the current mode.

The Y= Editor maintains an independent function list for each **Graph** mode setting. For example, suppose:

- In FUNCTION graphing mode, you define a set of \( y(x) \) functions. You change to POLAR graphing mode and define a set of \( r(\theta) \) equations.
- When you return to FUNCTION graphing mode, your \( y(x) \) functions are still defined in the Y= Editor. When you return to POLAR graphing mode, your \( r(\theta) \) equations are still defined.

### Selecting the Display Style

The **Above** and **Below** styles are not available for polar equations and are dimmed on the Y= Editor’s **Style** toolbar menu.

### Window Variables

The Window Editor maintains an independent set of Window variables for each **Graph** mode setting (just as the Y= Editor maintains independent function lists). Polar graphs use the following Window variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \theta_{\text{min}}, \theta_{\text{max}} )</td>
<td>Smallest and largest ( \theta ) values to evaluate.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| \( \theta_{\text{step}} \) | Increment for the \( \theta \) value. Polar equations are evaluated at:  
  \( r(\theta_{\text{min}}) \)  
  \( r(\theta_{\text{min}}+\theta_{\text{step}}) \)  
  \( r(\theta_{\text{min}}+2(\theta_{\text{step}})) \)  
  ... not to exceed ...  
  \( r(\theta_{\text{max}}) \) |
| \( x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}} \) | Boundaries of the viewing window.                                                                                                              |
| \( x_{\text{scl}}, y_{\text{scl}} \) | Distance between tick marks on the x and y axes.                                                                                              |

**Note:** You can use a negative \( \theta_{\text{step}} \). If so, \( \theta_{\text{min}} \) must be greater than \( \theta_{\text{max}} \).

Standard values (set when you select 6:ZoomStd from the [F2] Zoom toolbar menu) are:

<table>
<thead>
<tr>
<th>( \theta_{\text{min}} ) = 0.</th>
<th>( x_{\text{min}} = -10. )</th>
<th>( y_{\text{min}} = -10. )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \theta_{\text{max}} = 2\pi )</td>
<td>( (6.2831853... \text{ radians or 360 degrees}) )</td>
<td>( x_{\text{max}} = 10. )</td>
</tr>
<tr>
<td>( \theta_{\text{step}} = \pi/24 )</td>
<td>( (.1308996... \text{ radians or 7.5 degrees}) )</td>
<td>( x_{\text{scl}} = 1. )</td>
</tr>
</tbody>
</table>

You may need to change the standard values for the \( \theta \) variables (\( \theta_{\text{min}}, \theta_{\text{max}}, \theta_{\text{step}} \)) to ensure that enough points are plotted.

**Setting the Graph Format**

To display coordinates as \( r \) and \( \theta \) values, use:
to set Coordinates = POLAR. If Coordinates = RECT, the polar equations will be graphed properly, but coordinates will be displayed as x and y.

When you trace a polar equation, the θ coordinate is shown even if Coordinates = RECT.

**Exploring a Graph**

As in function graphing, you can explore a graph by using the following tools. Any displayed coordinates are shown in polar or rectangular form as set in the graph format.

<table>
<thead>
<tr>
<th>Tool</th>
<th>For Polar Graphs:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free-Moving Cursor</td>
<td>Works just as it does for function graphs.</td>
</tr>
<tr>
<td>F2 <strong>Zoom</strong></td>
<td>Works just as it does for function graphs.</td>
</tr>
<tr>
<td></td>
<td>• Only x (xmin, xmax, xscl) and y (ymin, ymax, yscl) Window variables are affected.</td>
</tr>
<tr>
<td></td>
<td>• The θ Window variables (θmin, θmax, θstep) are not affected unless you select 6:ZoomStd (which sets θmin = 0, θmax = 2π, and θstep = π/24).</td>
</tr>
</tbody>
</table>
During a trace, you can also evaluate \( r(\theta) \) by typing the \( \theta \) value and pressing \[ \text{Enter} \].

**Note:** You can use QuickCenter at any time during a trace, even if the cursor is still on the screen.
Parametric Graphing

Overview of Steps in Graphing Parametric Equations

To graph parametric equations, use the same general steps used for $y(x)$ functions as described in Basic Function Graphing. Any differences that apply to parametric equations are described on the following pages.

Graping Parametric Equations

1. Set Graph mode (MODE) to PARAMETRIC. Also set Angle mode, if necessary.

2. Define x and y components on Y= Editor (Y=).

3. Select (F4), which defined equations to graph. Select the x or y component, or both.

   **Note:** To turn off any stat data plots, press F5 5 or use F4 to deselect them.
4. Set the display style for an equation. You can set either the x or y component.

2nd [F6]

This is optional. For multiple equations, this helps visually distinguish one from another.

5. Define the viewing window ([WINDOW]).

F2 Zoom also changes the viewing window.

6. Change the graph format if necessary.

F1 9
– or –

7. Graph the selected equations ([GRAPH]).

Exploring the Graph

From the Graph screen, you can:

- Display the coordinates of any pixel by using the free-moving cursor, or of a plotted point by tracing a parametric equation.
• Use the F2 Zoom toolbar menu to zoom in or out on a portion of the graph.
• Use the F5 Math toolbar menu to find derivatives, tangents, etc. Some menu items are not available for parametric graphs.

**Differences in Parametric and Function Graphing**

This module assumes that you already know how to graph \(y(x)\) functions as described in Basic Function Graphing. This section describes the differences that apply to parametric equations.

**Setting the Graph Mode**

Use Mode to set Graph = PARAMETRIC before you define equations or set Window variables. The Y= Editor and the Window Editor let you enter information for the current Graph mode setting only.

**Defining Parametric Equations on the Y= Editor**

To graph a parametric equation, you must define both its x and y components. If you define only one component, the equation cannot be graphed. (However, you can use single components to generate an automatic table as described in Tables.)

Enter x and y components on separate lines.

You can define xt1(t) through xt99(t) and yt1(t) through yt99(t).
Be careful when using implied multiplication with t. For example:

<table>
<thead>
<tr>
<th>Enter:</th>
<th>Instead of:</th>
<th>Because:</th>
</tr>
</thead>
<tbody>
<tr>
<td>t*cos(60)</td>
<td>tcos(60)</td>
<td>tcos is interpreted as a user-defined function called tcos, not as implied multiplication. In most cases, this refers to a nonexistent function. So the TI-89 Titanium simply returns the function name, not a number.</td>
</tr>
</tbody>
</table>

**Note:** When using t, be sure implied multiplication is valid for your situation. You can use the Define command from the Home screen (see the Technical Reference module) to define functions and equations for any graphing mode, regardless of the current mode.

The Y= Editor maintains an independent function list for each Graph mode setting. For example, suppose:

- In FUNCTION graphing mode, you define a set of y(x) functions. You change to PARAMETRIC graphing mode and define a set of x and y components.
- When you return to FUNCTION graphing mode, your y(x) functions are still defined in the Y= Editor. When you return to PARAMETRIC graphing mode, your x and y components are still defined.

**Selecting Parametric Equations**

To graph a parametric equation, select either its x or y component or both. When you enter or edit a component, it is selected automatically.

Selecting x and y components separately can be useful for tables as described in Tables. With multiple parametric equations, you can select and compare all the x components or all the y components.
Selecting the Display Style

You can set the style for either the x or y component. For example, if you set the x component to **Dot**, the TI-89 Titanium automatically sets the y component to **Dot**.

**Note:** Use the **Animate** and **Path** styles for interesting projectile-motion effects.

The **Above** and **Below** styles are not available for parametric equations and are dimmed on the Y= Editor’s **Style** toolbar menu.

Window Variables

The Window Editor maintains an independent set of Window variables for each **Graph** mode setting (just as the Y= Editor maintains independent function lists). Parametric graphs use the following Window variables.

**Note:** You can use a negative **tstep**. If so, **tmin** must be greater than **tmax**.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tmin, tmax</strong></td>
<td>Smallest and largest <strong>t</strong> values to evaluate.</td>
</tr>
<tr>
<td><strong>tstep</strong></td>
<td>Increment for the <strong>t</strong> value. Parametric equations are evaluated at:</td>
</tr>
<tr>
<td></td>
<td>( x(t_{\text{min}}) \quad y(t_{\text{min}}) )</td>
</tr>
<tr>
<td></td>
<td>( x(t_{\text{min}} + t_{\text{step}}) \quad y(t_{\text{min}} + t_{\text{step}}) )</td>
</tr>
<tr>
<td></td>
<td>( x(t_{\text{min}} + 2(t_{\text{step}})) \quad y(t_{\text{min}} + 2(t_{\text{step}})) )</td>
</tr>
<tr>
<td></td>
<td>... not to exceed ... \quad ... not to exceed ...</td>
</tr>
<tr>
<td></td>
<td>( x(t_{\text{max}}) \quad y(t_{\text{max}}) )</td>
</tr>
<tr>
<td><strong>xmin, xmax</strong>, <strong>ymin, ymax</strong></td>
<td>Boundaries of the viewing window.</td>
</tr>
</tbody>
</table>
Standard values (set when you select 6:ZoomStd from the [F2] Zoom toolbar menu) are:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xscl, yscl</td>
<td>Distance between tick marks on the x and y axes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tmin</th>
<th>xmin</th>
<th>ymin</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-10.</td>
<td>-10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tmax</th>
<th>xmax</th>
<th>ymax</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2\pi$ (6.2831853... radians or 360 degrees)</td>
<td>10.</td>
<td>10.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tstep</th>
<th>xscl</th>
<th>yscl</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi/24$ (.1308996... radians or 7.5 degrees)</td>
<td>1.</td>
<td>1.</td>
</tr>
</tbody>
</table>

You may need to change the standard values for the $t$ variables ($t_{\text{min}}, t_{\text{max}}, t_{\text{step}}$) to ensure that enough points are plotted.

**Exploring a Graph**

As in function graphing, you can explore a graph by using the following tools.

**Note:** During a trace, you can also evaluate $x(t)$ and $y(t)$ by typing the $t$ value and pressing [ENTER]. You can use QuickCenter at any time during a trace, even if the cursor is still on the screen.

**Tool** | **For Parametric Graphs:**
---|---
Free-Moving Cursor | Works just as it does for function graphs.
<table>
<thead>
<tr>
<th>Tool</th>
<th>For Parametric Graphs:</th>
</tr>
</thead>
</table>
| **F2** Zoom | Works just as it does for function graphs, with the following exceptions:  
  - Only \(x\) (\(\text{xmin}, \text{xmax}, \text{xscl}\)) and \(y\) (\(\text{ymin}, \text{ymax}, \text{yscl}\)) Window variables are affected.  
  - The \(t\) Window variables (\(\text{tmin}, \text{tmax}, \text{tstep}\)) are not affected unless you select **6:ZoomStd** (which sets \(\text{tmin} = 0\), \(\text{tmax} = 2\pi\), and \(\text{tstep} = \pi/24\)). |
| **F3** Trace | Lets you move the cursor along a graph one \(\text{tstep}\) at a time.  
  - When you begin a trace, the cursor is on the first selected parametric equation at \(\text{tmin}\).  
  - QuickCenter applies to all directions. If you move the cursor off the screen (top or bottom, left or right), press **ENTER** to center the viewing window on the cursor location.  
  - Automatic panning is not available. If you move the cursor off the left or right side of the screen, the TI-89 Titanium will not automatically pan the viewing window. However, you can use QuickCenter. |
| **F5** Math | Only **1:Value**, **6:Derivatives**, **9:Distance**, **A:Tangent**, and **B:Arc** are available for parametric graphs. These tools are based on \(t\) values. For example:  
  - **1:Value** displays \(x\) and \(y\) values for a specified \(t\) value.  
  - **6:Derivatives** finds \(dy/dx\), \(dy/dt\), or \(dx/dt\) at a point defined for a specified \(t\) value. |
Sequence Graphing

Overview of Steps in Graphing Sequences

To graph sequences, use the same general steps used for $y(x)$ functions as described in Basic Function Graphing. Any differences are described on the following pages.

Graphing Sequences

1. Set Graph mode (MODE) to SEQUENCE. Also set Angle mode, if necessary.

2. Define sequences and, if needed, initial values on Y= Editor ([Y=]).

3. Select ([F4]) which defined sequences to graph. Do not select initial values.

   Note: To turn off any stat data plots, press [F5] 5 or use [F4] to deselect them.

4. Set the display style for a sequence.

   2nd [F6]

   For sequences, the default style is Square.
5. Define the viewing window ([WINDOW]).

**F2** Zoom also changes the viewing window.

6. Change the graph format if necessary.

**F1** 9
— or —

7. Graph the selected equations ([GRAPH]).

---

**Exploring the Graph**

From the Graph screen, you can:

- Display the coordinates of any pixel by using the free-moving cursor, or of a plotted point by tracing a sequence.
- Use the **F2** Zoom toolbar menu to zoom in or out on a portion of the graph.
- Use the **F5** Math toolbar menu to evaluate a sequence. Only **1:Value** is available for sequences.
- Plot sequences on Time (the default), Web, or Custom axes.

**Note:** You can also evaluate a sequence while tracing. Simply enter the n value directly from the keyboard.
Differences in Sequence and Function Graphing

This module assumes that you already know how to graph $y(x)$ functions as described in *Basic Function Graphing*. This section describes the differences that apply to sequences.

**Setting the Graph Mode**

Use \[\text{MODE}\] to set \textbf{Graph = SEQUENCE} before you define sequences or set Window variables. The Y= Editor and the Window Editor let you enter information for the current Graph mode setting only.

**Defining Sequences on the Y= Editor**

You can define sequences $u1(n)$ through $u99(n)$.

Use $ui$ only for recursive sequences, which require one or more initial values.

If a sequence requires more than one initial value, enter them as a list enclosed in braces \{ \} and separated by commas. You must use a list to enter two or more initial values.

Enter \{1,0\} even though \{1  0\} is shown in the sequence list.
If a sequence requires an initial value but you do not enter one, you will get an error when graphing.

On the Y= Editor, Axes lets you select the axes that are used to graph the sequences. Optionally, for sequences only, you can select different axes for the graph. TIME is the default.

<table>
<thead>
<tr>
<th>Axes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>Plots ( n ) on the ( x ) axis and ( u(n) ) on the ( y ) axis.</td>
</tr>
<tr>
<td>WEB</td>
<td>Plots ( u(n-1) ) on the ( x ) axis and ( u(n) ) on the ( y ) axis.</td>
</tr>
<tr>
<td>CUSTOM</td>
<td>Lets you select the ( x ) and ( y ) axes.</td>
</tr>
</tbody>
</table>

The Y= Editor maintains an independent function list for each Graph mode setting. For example, suppose:

- In FUNCTION graphing mode, you define a set of \( y(x) \) functions. You change to SEQUENCE graphing mode and define a set of \( u(n) \) sequences.
- When you return to FUNCTION graphing mode, your \( y(x) \) functions are still defined in the Y= Editor. When you return to SEQUENCE graphing mode, your \( u(n) \) sequences are still defined.

**Note:** You can use the Define command from the Home screen (see *Technical Reference*) to define functions and equations for any graphing mode, regardless of the current mode.
Selecting Sequences

With TIME and WEB axes, the TI-89 Titanium graphs only the selected sequences. If you entered any sequences that require an initial value, you must enter the corresponding ui value.

**Note:** With TIME and CUSTOM axes, all defined sequences are evaluated even if they are not plotted.

You can select a sequence

You cannot select its initial value.

With CUSTOM axes, when you specify a sequence in the custom settings, it is graphed regardless of whether it is selected.

Selecting the Display Style

Only the **Line**, **Dot**, **Square**, and **Thick** styles are available for sequence graphs. **Dot** and **Square** mark only those discrete integer values (in plotstep increments) at which a sequence is plotted.
# Window Variables

The Window Editor maintains an independent set of Window variables for each Graph mode setting (just as the Y= Editor maintains independent function lists). Sequence graphs use the following Window variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>nmin, nmax</strong></td>
<td>Smallest and largest ( n ) values to evaluate. Sequences are evaluated at: ( u(n_{\text{min}}) ), ( u(n_{\text{min}}+1) ), ( u(n_{\text{min}}+2) ), ... not to exceed ... ( u(n_{\text{max}}) )</td>
</tr>
<tr>
<td><strong>plotStrt</strong></td>
<td>The term number that will be the first one plotted (depending on plotstep). For example, to begin plotting with the 2nd term in the sequence, set <strong>plotstrt = 2</strong>. The first term will be evaluated at <strong>nmin</strong> but not plotted.</td>
</tr>
<tr>
<td><strong>plotStep</strong></td>
<td>Incremental ( n ) value for graphing only. This does not affect how the sequence is evaluated, only which points are plotted. For example, suppose <strong>plotstep = 2</strong>. The sequence is evaluated at each consecutive integer but is plotted at only every other integer.</td>
</tr>
<tr>
<td><strong>xmin, xmax, ymin, ymax</strong></td>
<td>Boundaries of the viewing window.</td>
</tr>
<tr>
<td><strong>xscl, yscl</strong></td>
<td>Distance between tick marks on the x and y axes.</td>
</tr>
</tbody>
</table>

**Note:** Both **nmin** and **nmax** must be positive integers, although **nmin** can be zero; **nmin**, **nmax**, **plotstrt** and **plotstep** must be integers \( \geq 1 \). If you do not enter integers, they will be rounded to integers.
Standard values (set when you select 6:ZoomStd from the [F2] Zoom toolbar menu) are:

nmin = 1.        xmin = -10.         ymin = -10.
plotstrt = 1.     xscl = 1.         yscl = 1.
plotstep = 1.

You may need to change the standard values for the n and plot variables to ensure that sufficient points are plotted.

To see how plotstrt affects graph, look at the following examples of a recursive sequence.

This graph is plotted beginning with the 1st term.

```
plotStrt=1
```

This graph is plotted beginning with the 9th term.

```
plotStrt=9
```

**Note:** Both of these graphs use the same Window variables, except for plotstrt.
With TIME axes (from Axes on the Y= Editor), you can set $\text{plotstrt} = 1$ and still graph only a selected part of the sequence. Simply define a viewing window that shows only the area of the coordinate plane you want to view.

You could set:

- $x_{\text{min}}$ = first $n$ value to graph
- $x_{\text{max}}$ = $n_{\text{max}}$ (although you can use other values)
- $y_{\text{min}}$ and $y_{\text{max}}$ = expected values for the sequence

### Changing the Graph Format

The Graph Order format is not available.

- With TIME or CUSTOM axes, multiple sequences are always plotted simultaneously.
- With WEB axes, multiple sequences are always plotted sequentially.

### Exploring a Graph

As in function graphing, you can explore a graph by using the following tools. Any displayed coordinates are shown in rectangular or polar form as set in the graph format.

<table>
<thead>
<tr>
<th>Tool</th>
<th>For Sequence Graphs:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free-Moving Cursor</td>
<td>Works just as it does for function graphs.</td>
</tr>
</tbody>
</table>
### Tool For Sequence Graphs:

**F2 Zoom**
- Works just as it does for function graphs.
  - Only \( x (\text{xmin}, \text{xmax}, \text{xscl}) \) and \( y (\text{ymin}, \text{ymax}, \text{yscl}) \) Window variables are affected.
  - The \( n \) and plot Window variables (\( \text{nmin}, \text{nmax}, \text{plotStrt}, \text{plotStep} \)) are not affected unless you select 6:ZoomStd (which sets all Window variables to their standard values).

**F3 Trace**
- Depending on whether you use TIME, CUSTOM, or WEB axes, Trace operates very differently.
  - With TIME or CUSTOM axes, you move the cursor along the sequence one plotstep at a time. To move approximately ten plotted points at a time, press \( \text{2nd} \) → or \( \text{2nd} \) ←.
    - When you begin a trace, the cursor is on the first selected sequence at the term number specified by plotStrt, even if it is outside the viewing window.
    - QuickCenter applies to all directions. If you move the cursor off the screen (top or bottom, left or right), press \( \text{ENTER} \) to center the viewing window on the cursor location.
  - With WEB axes, the trace cursor follows the web, not the sequence.

**F5 Math**
- Only 1:Value is available for sequence graphs.
  - With TIME and WEB axes, the \( u(n) \) value (represented by \( y_c \)) is displayed for a specified \( n \) value.
  - With CUSTOM axes, the values that correspond to \( x \) and \( y \) depend on the axes you choose.
During a trace, you can evaluate a sequence by typing a value for \( n \) and pressing [\text{ENTER}]. You can use QuickCenter at any time during a trace, even if the cursor is still on the screen.

## Setting Axes for Time, Web, or Custom Plots

For sequences only, you can select different types of axes for the graph. Examples of the different types are given later in this module.

### Displaying the AXES Dialog Box

From the Y= Editor, Axes:

- Depending on the current Axes setting, some items may be dimmed.
- To exit without making any changes, press [\text{ESC}].

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Axes**      | **TIME** — Plots \( u(n) \) on the y axis and \( n \) on the x axis.  
WEB — Plots \( u(n) \) on the y axis and \( u(n-1) \) on the x axis.  
CUSTOM — Lets you select the x and y axes. |
| **Build Web** | Active only when Axes = WEB, this specifies whether a web is drawn manually (TRACE) or automatically (AUTO). |
To change any of these settings, use the same procedure that you use to change other types of dialog boxes, such as the MODE dialog box.

**Using Web Plots**

A web plot graphs $u(n)$ vs. $u(n-1)$, which lets you study the long-term behavior of a recursive sequence. The examples in this section also show how the initial value can affect a sequence’s behavior.

**Valid Functions for Web Plots**

A sequence must meet the following criteria; otherwise, it will not be graphed properly on WEB axes. The sequence:

- Must be recursive with only one recursion level; $u(n-1)$ but not $u(n-2)$.
- Cannot reference $n$ directly.
- Cannot reference any other defined sequence except itself.

**When You Display the Graph Screen**

After you select WEB axes and display the Graph screen, the TI-89 Titanium:
• Draws a \( y=x \) reference line.
• Plots the selected sequence definitions as functions, with \( u(n-1) \) as the independent variable. This effectively converts a recursive sequence into a nonrecursive form for graphing.

For example, consider the sequence \( u_1(n) = \sqrt{5 - u_1(n-1)} \) and an initial value of \( u_{i1}=1 \). The TI-89 Titanium draws the \( y=x \) reference line and then plots \( y = y = \sqrt{5-x} \).

**Drawing the Web**

After the sequence is plotted, the web may be displayed manually or automatically, depending on how you set **Build Web** on the AXES dialog box.

<table>
<thead>
<tr>
<th>If Build Web =</th>
<th>The web is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACE</td>
<td>Not drawn until you press ([F3]). The web is then drawn step-by-step as you move the trace cursor (you must have an initial value before using Trace).</td>
</tr>
<tr>
<td>AUTO</td>
<td>Drawn automatically. You can then press ([F3]) to trace the web and display its coordinates.</td>
</tr>
</tbody>
</table>

**Note:** With WEB axes, you cannot trace along the sequence itself as you do in other graphing modes.

The web:

1. Starts on the x axis at the initial value \( u_i \) (when \( \text{plotstr} = 1 \)).
2. Moves vertically (either up or down) to the sequence.
3. Moves horizontally to the \( y=x \) reference line.
4. Repeats this vertical and horizontal movement until \( n=n_{\text{max}} \).

**Note:** The web starts at \( \text{plotstrt} \). The value of \( n \) is incremented by 1 each time the web moves to the sequence (\( \text{plotStep} \) is ignored).

**Example: Convergence**

1. On the Y= Editor (\( [Y=] \)), define \( u_1(n) = -.8u_1(n-1) + 3.6 \). Set initial value \( u_{i1} = -4 \).
2. Set \( \text{Axes} = \text{TIME} \).
3. On the Window Editor (\( [\text{WINDOW}] \)), set the Window variables.

\[
\begin{align*}
\text{nmin} &= 1 \\
\text{nmax} &= 25 \\
\text{plotstrt} &= 1 \\
\text{plotstep} &= 1 \\
\text{xmin} &= 0 \\
\text{xmax} &= 25 \\
\text{ymin} &= -10 \\
\text{ymax} &= 10 \\
\text{xsc} &= 1 \\
\text{ysc} &= 1
\end{align*}
\]

4. Graph the sequence (\( [\text{GRAPH}] \)). By default, a sequence uses the **Square** display style.

5. On the Y= Editor, set \( \text{Axes} = \text{WEB} \) and \( \text{Build Web} = \text{AUTO} \).
6. On the Window Editor, change the Window variables.

\[
\begin{align*}
\text{nmin} &= 1 & \text{xmin} &= -10 & \text{ymin} &= -10 \\
\text{nmax} &= 25 & \text{xmax} &= 10 & \text{ymax} &= 10 \\
\text{plotstrt} &= 1 & \text{xsc} &= 1 & \text{ysc} &= 1
\end{align*}
\]

7. Regraph the sequence.

Web plots are always shown as lines, regardless of the selected display style.

**Note:** During a trace, you can move the cursor to a specified \( n \) value by typing the value and pressing [ENTER].

8. Press [F3]. As you press \( \uparrow \), the trace cursor follows the web. The screen displays the cursor coordinates \( nc, xc, \) and \( yc \) (where \( xc \) and \( yc \) represent \( u(n-1) \) and \( u(n) \), respectively).

As you trace to larger values of \( nc \), you can see \( xc \) and \( yc \) approach the convergence point.

**Note:** When the \( nc \) value changes, the cursor is on the sequence. The next time you press \( \uparrow \), \( nc \) stays the same but the cursor is now on the \( y=x \) reference line.
**Example: Divergence**

1. On the Y= Editor ([Y=]), define \( u_1(n) = 3.2u_1(n-1) - .8(u_1(n-1))^2 \). Set initial value \( u_{i1} = 4.45 \).

2. Set *Axes = TIME*.

3. On the Window Editor ([WINDOW]), set the Window variables.

   - \( n_{\text{min}} = 0 \)
   - \( n_{\text{max}} = 10 \)
   - \( \text{plotstrt} = 1 \)
   - \( \text{plotstep} = 1 \)
   - \( x_{\text{min}} = 0 \)
   - \( x_{\text{max}} = 10 \)
   - \( \text{xscl} = 1 \)
   - \( y_{\text{min}} = -75 \)
   - \( y_{\text{max}} = 10 \)
   - \( \text{yscl} = 1 \)

4. Graph the sequence ([GRAPH]).

   Because the sequence quickly diverges to large negative values, only a few points are plotted.

5. On the Y= Editor, set *Axes = WEB* and *Build Web = AUTO*.

6. On the Window Editor ([WINDOW]), set the Window variables.

   - \( n_{\text{min}} = 0 \)
   - \( n_{\text{max}} = 10 \)
   - \( \text{plotstrt} = 1 \)
   - \( \text{plotstep} = 1 \)
   - \( x_{\text{min}} = -10 \)
   - \( x_{\text{max}} = 10 \)
   - \( \text{xscl} = 1 \)
   - \( y_{\text{min}} = -10 \)
   - \( y_{\text{max}} = 10 \)
   - \( \text{yscl} = 1 \)
Sequence Graphing

Example: Oscillation

This example shows how the initial value can affect a sequence.

1. On the Y= Editor (Y=), use the same sequence defined in the divergence example: \( u_1(n) = 3.2u_1(n-1) - .8(u_1(n-1))^2 \). Set initial value \( u_{i1} = 0.5 \).

2. Set Axes = TIME.

3. On the Window Editor ([WINDOW]), set the Window variables.

   \[
   \begin{align*}
   n_{\text{min}} &= 1 \\
   n_{\text{max}} &= 100 \\
   \text{plotstr} &= 1 \\
   \text{plotstep} &= 1 \\
   x_{\text{min}} &= 0 \\
   x_{\text{max}} &= 100 \\
   x_{\text{scl}} &= 10 \\
   y_{\text{min}} &= 0 \\
   y_{\text{max}} &= 5 \\
   y_{\text{scl}} &= 1
   \end{align*}
   \]

4. Graph the sequence ([GRAPH]).

   \[
   \begin{align*}
   u(n)
   \end{align*}
   \]

   **Note:** Compare this graph with the divergence example. This is the same sequence with a different initial value.

5. On the Y= Editor, set Axes = WEB and Build Web = AUTO.
6. On the Window Editor ([WINDOW]), set the Window variables.

\[
\begin{align*}
\text{nmin} &= 1 & \text{xmin} &= 2.68 & \text{ymin} &= 4.7 \\
\text{nmax} &= 100 & \text{xmax} &= 6.47 & \text{ymax} &= 47 \\
\text{plotstrt} &= 1 & \text{xscl} &= 1 & \text{yscl} &= 1
\end{align*}
\]

7. Regraph the sequence.

**Note:** The web moves to an orbit oscillating between two stable points.

8. Press [F3]. Then use \( \odot \) to trace the web.

As you trace to larger values of \( n \), notice that \( x_c \) and \( y_c \) oscillate between 2.05218 and 3.19782.

9. On the Window Editor, set \text{plotstrt}=50. Then regraph the sequence.

**Note:** By starting the web plot at a later term, the stable oscillation orbit is shown more clearly.

### Using Custom Plots

CUSTOM axes give you great flexibility in graphing sequences. As shown in the following example, CUSTOM axes are particularly effective for showing relationships between one sequence and another.
**Example: Predator-Prey Model**

Using the predator-prey model in biology, determine the numbers of rabbits and foxes that maintain population equilibrium in a certain region.

\[ R = \text{Number of rabbits} \]
\[ M = \text{Growth rate of rabbits if there are no foxes (use .05)} \]
\[ K = \text{Rate at which foxes can kill rabbits (use .001)} \]
\[ W = \text{Number of foxes} \]
\[ G = \text{Growth rate of foxes if there are rabbits (use .0002)} \]
\[ D = \text{Death rate of foxes if there are no rabbits (use .03)} \]

\[ R_n = R_{n-1} (1 + M - K W_{n-1}) \]
\[ W_n = W_{n-1} (1 + G R_{n-1} - D) \]

1. On the Y= Editor (\[ Y= \]), define the sequences and initial values for \( R_n \) and \( W_n \).
   \[ u1(n) = u1(n-1) * (1 + .05 - .001 * u2(n-1)) \]
   \[ u1 = 200 \]
   \[ u2(n) = u2(n-1) * (1 + .0002 * u1(n-1) - .03) \]
   \[ u2 = 50 \]

   **Note:** Assume there are initially 200 rabbits and 50 foxes.

2. Set Axes = TIME.
3. On the Window Editor ([WINDOW]), set the Window variables.

\[
\begin{align*}
\text{nmin}=0 & \quad \text{xmin}=0 & \quad \text{ymin}=0 \\
\text{nmax}=400 & \quad \text{xmax}=400 & \quad \text{ymax}=300 \\
\text{plotstrt}=1 & \quad \text{xscl}=100 & \quad \text{yscl}=100 \\
\text{plotstep}=1 & \\
\end{align*}
\]

4. Graph the sequence ([GRAPH]).

\[
\text{Note: Use } \text{F3} \text{ to individually trace the number of rabbits } u_1(n) \text{ and foxes } u_2(n) \text{ over time } (n).
\]

5. On the Y= Editor, set \textbf{Axes = CUSTOM}, \textbf{X Axis = u1}, and \textbf{Y Axis = u2}.

6. On the Window Editor ([WINDOW]), set the Window variables.

\[
\begin{align*}
\text{nmin}=0 & \quad \text{xmin}=84 & \quad \text{ymin}=25 \\
\text{nmax}=400 & \quad \text{xmax}=237 & \quad \text{ymax}=75 \\
\text{plotstrt}=1 & \quad \text{xscl}=50 & \quad \text{yscl}=10 \\
\text{plotstep}=1 & \\
\end{align*}
\]

7. Regraph the sequence.

\[
\text{Note: Use } \text{F3} \text{ to trace both the number of rabbits (xc) and foxes (yc) over the cycle of 400 generations.}
\]
Using a Sequence to Generate a Table

Previous sections described how to graph a sequence. You can also use a sequence to generate a table. Refer to *Tables* for detailed information.

**Example: Fibonacci Sequence**

In a Fibonacci sequence, the first two terms are 1 and 1. Each succeeding term is the sum of the two immediately preceding terms.

1. On the Y= Editor ([Y=]), define the sequence and set the initial values as shown.

   ![Sequence List]

   You must enter \(\{1,1\}\), although \(\{1\ 1\}\) is shown in the sequence list.

2. Set table parameters ([TBLSET]) to:
   - \(\text{tblStart} = 1\)
   - \(\Delta \text{tbl} = 1\)
   - \(\text{Independent} = \text{AUTO}\)

   ![Table Setup]

   This item is dimmed if you are not using TIME axes.
3. Set Window variables ([WINDOW]) so that `nmin` has the same value as `tblStart`.

```
min=3
max=10.
plotStart=1.
plotStep=1.
xmin=-10.
xmax=10.
xsc1=1.
ymin=-10.
ymax=10.
ysc1=1.
```

4. Display the table ([TABLE]).

```
<table>
<thead>
<tr>
<th>n</th>
<th>F1</th>
<th>F2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>
```

Fibonacci sequence is in column 2.

5. Scroll down the table ( or 2nd ) to see more of the sequence.
3D Graphing

Overview of Steps in Graphing 3D Equations

To graph 3D equations, use the same general steps used for \( y(x) \) functions as described in *Basic Function Graphing*. Any differences that apply to 3D equations are described on the following pages.

**Graphing 3D Equations**

1. Set Graph mode (MODE) to 3D. Also set Angle mode, if necessary.

2. Define 3D equations on Y= Editor ([Y=]).

3. Select (F4) which equation to graph. You can select only one 3D equation.
   To turn off any stat data plots, press F5 5 or use F4 to deselect them.

4. Define the viewing cube ([WINDOW]).
   For 3D graphs, the viewing window is called the viewing cube. F2 Zoom also changes the viewing cube.
5. Change the graph format if necessary.

F1 9
– or -
♦ I

Note: To help you see the orientation of 3D graphs, turn on Axes and Labels.

6. Graph the selected equations
(♦ [GRAPH]).

Note: Before displaying the graph, the screen shows the “percent evaluated.”

Exploring the Graph

From the Graph screen, you can:

• Trace the equation.
• Use the F2 Zoom toolbar menu to zoom in or out on a portion of the graph. Some of the menu items are dimmed because they are not available for 3D graphs.
• Use the F5 Math toolbar menu to evaluate the equation at a specified point. Only 1:Value is available for 3D graphs.

You can also evaluate \( z(x, y) \) while tracing. Type the x value and press \( \text{ENTER} \); then type the y value and press \( \text{ENTER} \).
Differences in 3D and Function Graphing

This module assumes that you already know how to graph $y(x)$ functions as described in Basic Function Graphing. This section describes the differences that apply to 3D equations.

Setting the Graph Mode

Use MODE to set Graph = 3D before you define equations or set Window variables. The Y= Editor and the Window Editor let you enter information for the current Graph mode setting only.

Defining 3D Equations on the Y= Editor

The Y= Editor maintains an independent function list for each Graph mode setting. For example, suppose:

- In FUNCTION graphing mode, you define a set of $y(x)$ functions. You change to 3D graphing mode and define a set of $z(x,y)$ equations.
• When you return to FUNCTION graphing mode, your $y(x)$ functions are still defined in the Y= Editor. When you return to 3D graphing mode, your $z(x,y)$ equations are still defined.

**Note:** You can use the Define command from the Home screen (see the Technical Reference module) to define functions and equations for any graphing mode, regardless of the current mode.

### Selecting the Display Style

Because you can graph only one 3D equation at a time, display styles are not available. On the Y= Editor, the Style toolbar menu is dimmed.

For 3D equations, however, you can use:

- or -

  $\text{F1 9}$

  $\text{F2 T}$

to set the Style format to WIRE FRAME or HIDDEN SURFACE.

### Window Variables

The Window Editor maintains an independent set of Window variables for each Graph mode setting (just as the Y= Editor maintains independent function lists). 3D graphs use the following Window variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{eye}<em>\theta$, $\text{eye}</em>\phi$, $\text{eye}_\psi$</td>
<td>Angles (always in degrees) used to view the graph.</td>
</tr>
</tbody>
</table>
### Variable Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{min}$, $x_{max}$, $y_{min}$, $y_{max}$, $z_{min}$, $z_{max}$</td>
<td>Boundaries of the viewing cube.</td>
</tr>
<tr>
<td>$x_{grid}$, $y_{grid}$</td>
<td>The distance between $x_{min}$ and $x_{max}$ and between $y_{min}$ and $y_{max}$ is divided into the specified number of grids. The $z(x,y)$ equation is evaluated at each grid point where the grid lines (or grid wires) intersect. The incremental value along $x$ and $y$ is calculated as: $x$ increment = $\frac{x_{max} - x_{min}}{x_{grid}}$ and $y$ increment = $\frac{y_{max} - y_{min}}{y_{grid}}$. The number of grid wires is $x_{grid} + 1$ and $y_{grid} + 1$. For example, when $x_{grid} = 14$ and $y_{grid} = 14$, the xy grid consists of 225 (15 $\times$ 15) grid points.</td>
</tr>
<tr>
<td>$n_{contour}$</td>
<td>The number of contours evenly distributed along the displayed range of $z$ values.</td>
</tr>
</tbody>
</table>
**Note:** If you enter a fractional number for \( xgrid \) or \( ygrid \), it is rounded to the nearest whole number \( \geq 1 \). The 3D mode does not have \( scl \) Window variables, so you cannot set tick marks on the axes.

Standard values (set when you select 6:ZoomStd from the [F2] Zoom toolbar menu) are:

\[
\begin{align*}
\text{eye}\theta &= 20. \\
\text{eye}\phi &= 70. \\
\text{eye}\psi &= 0. \\
\text{xmin} &= -10. \\
\text{xmax} &= 10. \\
\text{xgrid} &= 14. \\
\text{ymin} &= -10. \\
\text{ymax} &= 10. \\
\text{ygrid} &= 14. \\
\text{zmin} &= -10. \\
\text{zmax} &= 10. \\
\text{ncontour} &= 5.
\end{align*}
\]

You may need to increase the standard values for the grid variables (\( xgrid, ygrid \)) to ensure that enough points are plotted.

**Note:** Increasing the grid variables decreases the graphing speed.

**Setting the Graph Format**

The Axes and Style formats are specific to the 3D graphing mode.

**Exploring a Graph**

As in function graphing, you can explore a graph by using the following tools. Any displayed coordinates are shown in rectangular or cylindrical form as set in the graph format. In 3D graphing, cylindrical coordinates are shown when you use use: 

[F1] 9

– or –
to set Coordinates = POLAR.

<table>
<thead>
<tr>
<th>Tool</th>
<th>For 3D Graphs:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free-Moving Cursor</td>
<td>The free-moving cursor is not available.</td>
</tr>
<tr>
<td><strong>F2 Zoom</strong></td>
<td>Works essentially the same as it does for function graphs, but remember that you are now using three dimensions instead of two.</td>
</tr>
<tr>
<td></td>
<td>• Only the following zooms are available:</td>
</tr>
<tr>
<td></td>
<td>2:ZoomIn, 3:ZoomOut, 5:ZoomSqr, 6:ZoomStd, A:ZoomFit, B:Memory, C:SetFactors</td>
</tr>
<tr>
<td></td>
<td>• Only x (xmin, xmax), y (ymin, ymax), and z (zmin, zmax) Window variables are affected.</td>
</tr>
<tr>
<td></td>
<td>• The grid (xgrid, ygrid) and eye (eyeθ, eyeϕ, eyeψ) Window variables are not affected unless you select 6:ZoomStd (which resets these variables to their standard values).</td>
</tr>
<tr>
<td><strong>F3 Trace</strong></td>
<td>Lets you move the cursor along a grid wire from one grid point to the next on the 3D surface.</td>
</tr>
<tr>
<td></td>
<td>• When you begin a trace, the cursor appears at the midpoint of the xy grid.</td>
</tr>
<tr>
<td></td>
<td>• QuickCenter is available. At any time during a trace, regardless of the cursor’s location, you can press [ENTER] to center the viewing cube on the cursor.</td>
</tr>
<tr>
<td></td>
<td>• Cursor movement is restricted in the x and y directions. You cannot move the cursor beyond the viewing cube boundaries set by xmin, xmax, ymin, and ymax.</td>
</tr>
</tbody>
</table>
**F5 Math**  
Only 1:Value is available for 3D graphs. This tool displays the z value for a specified x and y value. 
After selecting **1:Value**, type the x value and press [ENTER]. Then type the y value and press [ENTER].

---

**Note:** During a trace, you can also evaluate \( z(x,y) \). Type the x value and press [ENTER]; then type the y value and press [ENTER].

---

**Moving the Cursor in 3D**

When you move the cursor along a 3D surface, it may not be obvious why the cursor moves as it does. 3D graphs have two independent variables \((x,y)\) instead of one, and the x and y axes have a different orientation than other graphing modes.

---

**How to Move the Cursor**

On a 3D surface, the cursor always follows along a grid wire.

<table>
<thead>
<tr>
<th>Cursor Key</th>
<th>Moves the cursor to the next grid point in the:</th>
</tr>
</thead>
<tbody>
<tr>
<td>⏳</td>
<td>Positive x direction</td>
</tr>
<tr>
<td>⏪</td>
<td>Negative x direction</td>
</tr>
<tr>
<td>▲</td>
<td>Positive y direction</td>
</tr>
<tr>
<td>▼</td>
<td>Negative y direction</td>
</tr>
</tbody>
</table>
**Note:** You can move the cursor only within the x and y boundaries set by Window variables `xmin`, `xmax`, `ymin`, and `ymax`.

Although the rules are straightforward, the actual cursor movement can be confusing unless you know the orientation of the axes.

In 2D graphing, the x and y axes always have the same orientation relative to the Graph screen.

In 3D graphing, x and y have a different orientation relative to the Graph screen. Also, you can rotate and/or elevate the viewing angle.

\[
\text{eye} \theta = 20 \quad \text{eye} \phi = 70 \quad \text{eye} \psi = 0
\]

**Note:** To show the axes and their labels from the Y= Editor, Window Editor, or Graph screen, use:

Simple Example of Moving the Cursor

The following graph shows a sloped plane that has the equation \( z_1(x,y) = -(x + y) / 2 \). Suppose you want to trace around the displayed boundary.
When you press \texttt{F3}, the trace cursor appears at the midpoint of the xy grid. Use the cursor pad to move the cursor to any edge.

- ⬆️ moves in a positive x direction, up to xmax.
- ⬇️ moves in a negative y direction, back to ymin.
- ➡️ moves in a positive y direction, up to ymax.
- ⬅️ moves in a negative x direction, back to xmin.

By displaying and labeling the axes, you can more easily see the pattern in the cursor movement. To move grid points closer together, you can increase Window variables \texttt{xgrid} and \texttt{ygrid}.

When the trace cursor is on an interior point in the displayed plane, the cursor moves from one grid point to the next along one of the grid wires. You cannot move diagonally across the grid. Notice that the grid wires may not appear parallel to the axes.

**Example of the Cursor on a Hidden Surface**

On more complex shapes, the cursor may appear as if it is not on a grid point. This is an optical illusion caused when the cursor is on a hidden surface.
For example, consider a saddle shape $z_1(x,y) = \frac{(x^2-y^2)}{3}$. The following graph shows the view looking down the y axis.

Now look at the same shape at $10^\circ$ from the x axis ($\text{eye}\theta = 10$).

Note: To cut away the front of the saddle in this example, set $\text{xmax}=0$ to show only negative x values.

**Example of an “Off the Curve” Cursor**

Although the cursor can move only along a grid wire, you will see many cases where the cursor does not appear to be on the 3D surface at all. This occurs when the z axis is too short to show $z(x,y)$ for the corresponding x and y values.
For example, suppose you trace the paraboloid \( z(x,y) = x^2 + .5y^2 \) graphed with the indicated Window variables. You can easily move the cursor to a position such as:

\[
\begin{align*}
\text{xc} & : 4.28571 \\
\text{yc} & : -4.28571 \\
\text{zc} & : 27.551
\end{align*}
\]

Although the cursor is actually tracing the paraboloid, it appears off the curve because the trace coordinates:

- \( \text{xc} \) and \( \text{yc} \) are within the viewing cube.
- but
- \( \text{zc} \) is outside the viewing cube.

**Note:** QuickCenter lets you center the viewing cube on the cursor’s location. Simply press \( \text{ENTER} \).

When \( \text{zc} \) is outside the z boundary of the viewing cube, the cursor is physically displayed at \( \text{zmin} \) or \( \text{zmax} \) (although the screen shows the correct trace coordinates).

### Rotating and/or Elevating the Viewing Angle

In 3D graphing mode, the \( \text{eye} \theta \) and \( \text{eye} \phi \) Window variables let you set viewing angles that determine your line of sight. The \( \text{eye} \psi \) Window variable lets you rotate the graph around that line of sight.
How the Viewing Angle Is Measured

The viewing angle has three components:

- **eye\(\theta\)** — angle in degrees from the positive x axis.
- **eye\(\phi\)** — angle in degrees from the positive z axis.
- **eye\(\psi\)** — angle in degrees by which the graph is rotated counter-clockwise around the line of sight set by **eye\(\theta\)** and **eye\(\phi\)**.

**Note**: When **eye\(\psi\)** = 0, the z axis is vertical on the screen. When **eye\(\psi\)** = 90, the z axis is rotated 90° counterclockwise and is horizontal.

In the Window Editor ([WINDOW]), always enter **eye\(\theta\)**, **eye\(\phi\)**, and **eye\(\psi\)** in degrees, regardless of the current angle mode.

Do not enter a ° symbol. For example, type 20, 70, and 0, not 20°, 70° and 0°.
**Effect of Changing \(\text{eye}\theta\)**

The view on the Graph screen is always oriented along the viewing angle. From this point of view, you can change \(\text{eye}\theta\) to rotate the viewing angle around the \(z\) axis.

\[
z_1(x,y) = \frac{x^3y - y^3x}{390}
\]

*In this example \(\text{eye}\phi = 70\)*

<table>
<thead>
<tr>
<th>(\text{eye}\theta = 20)</th>
<th>(\text{eye}\theta = 50)</th>
<th>(\text{eye}\theta = 80)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram 1" /></td>
<td><img src="image2.png" alt="Diagram 2" /></td>
<td><img src="image3.png" alt="Diagram 3" /></td>
</tr>
</tbody>
</table>

**Note:** This example increments \(\text{eye}\theta\) by 30.
### Effect of Changing $\text{eye}_\phi$ phi

By changing $\text{eye}_\phi$, you can elevate your viewing angle above the xy plane. If $90 < \text{eye}_\phi < 270$, the viewing angle is below the xy plane.

\[
z_1(x,y) = \frac{(x^3 y - y^3 x)}{390}
\]

In this example $\text{eye}_\theta = 20$

<table>
<thead>
<tr>
<th>$\text{eye}_\phi$</th>
<th>Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>![Graph with $\text{eye}_\phi = 90$]</td>
</tr>
<tr>
<td>70</td>
<td>![Graph with $\text{eye}_\phi = 70$]</td>
</tr>
<tr>
<td>50</td>
<td>![Graph with $\text{eye}_\phi = 50$]</td>
</tr>
</tbody>
</table>

**Note:** This example starts on the xy plane ($\text{eye}_\phi = 90$) and decrements $\text{eye}_\phi$ by 20 to elevate the viewing angle.

### Effect of Changing $\text{eye}_\psi$ psi

The view on the Graph screen is always oriented along the viewing angles set by $\text{eye}_\theta$ and $\text{eye}_\phi$. You can change $\text{eye}_\psi$ to rotate the graph around that line of sight.
**Note:** During rotation, the axes expand or contract to fit the screen’s width and height. This causes some distortion as shown in the example.

\[
z_1(x, y) = (x^3y - y^3x) / 390
\]

In this example, \(\text{eye}\theta=20\) and \(\text{eye}\phi=70\)

<table>
<thead>
<tr>
<th>(\text{eye}\psi)</th>
<th>(\text{eye}\psi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(z=10)</td>
</tr>
<tr>
<td>45</td>
<td>(z=10)</td>
</tr>
<tr>
<td>90</td>
<td>(z=10)</td>
</tr>
</tbody>
</table>

When \(\text{eye}\psi=0\), the \(z\) axis runs the height of the screen.

When \(\text{eye}\psi=90\), the \(z\) axis runs the width of the screen.
As the z axis rotates 90°, its range (−10 to 10 in this example) expands to almost twice its original length. Likewise, the x and y axes expand or contract.

**From the Home Screen or a Program**

The **eye** values are stored in the system variables **eye**θ, **eye**φ, and **eye**ψ. You can access or store to these variables as necessary.

- To type φ or ψ, press ▼ [alpha] [F] or ▼ [Y], respectively. You can also press 2nd [CHAR] and use the Greek menu.

**Animating a 3D Graph Interactively**

After plotting any 3D graph, you can change the viewing angle interactively by using the cursor.

**The Viewing Orbit**

When using ◀ and ▶ to animate a graph, think of it as moving the viewing angle along its “viewing orbit” around the graph.

Moving along this orbit can cause the z axis to wobble slightly during the animation.
**Note:** The viewing orbit affects the eye Window variables in differing amounts.

### Animating the Graph

<table>
<thead>
<tr>
<th>To:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Animate the graph incrementally.</td>
<td>Press and release the cursor quickly.</td>
</tr>
<tr>
<td>Move along the viewing orbit.</td>
<td>◀ or ▼</td>
</tr>
<tr>
<td>Change the viewing orbit’s elevation. (primarily increases or decreases eyeφ)</td>
<td>◡ or ◥</td>
</tr>
<tr>
<td>Animate the graph continuously.</td>
<td>Press and hold the cursor for about 1 second, and then release it.</td>
</tr>
<tr>
<td></td>
<td>To stop, press [ESC], [ENTER], [ON], or [ ] (space).</td>
</tr>
<tr>
<td>Change between 4 animation speeds (increase or decrease the incremental changes in the eye Window variables).</td>
<td>Press [+ or [-].</td>
</tr>
<tr>
<td>Change the viewing angle of a non-animated graph to look along the x, y, or z axis.</td>
<td>Press X, Y or Z, respectively.</td>
</tr>
<tr>
<td>Return to the initial eye angle values.</td>
<td>Press 0 (zero).</td>
</tr>
</tbody>
</table>

**Notes:** If the graph is shown in expanded view, it returns to normal view automatically when you press a cursor key.
• After animating the graph, you can stop and then re-start the animation in the same
direction by pressing: 

  \[ \text{ENTER} \text{ or } \text{alpha } [ \text{~} ] \]

• During an animation, you can switch to the next graph format style by pressing:

  \[ 1 \]

• You can view a graphic that shows the eye angles.

### Animating a Series of Graph Pictures

You can also animate a graph by saving a series of graph pictures and then flipping (or
cycling) through those pictures. Refer to “Animating a Series of Graph Pictures”
Additonal Graphing Topics. This method gives you more control over the Window
variable values, particularly \( \text{eye} \psi \), which rotates the graph.

### Changing the Axes and Style Formats

With its default settings, the TI-89 Titanium displays hidden surfaces on a 3D graph but
does not display the axes. However, you can change the graph format at any time.

### Displaying the GRAPH FORMATS Dialog Box

From the Y= Editor, Window Editor, or Graph screen, press:

\[ \text{F1} 9 \]
- or -

- The dialog box shows the current graph format settings.
- To exit without making a change, press \[\text{ESC}\].

To change any of these settings, use the same procedure that you use to change other types of dialog boxes, such as the MODE dialog box.

**Examples of Axes Settings**

To display the valid **Axes** settings, highlight the current setting and press \[\text{Esc}\].

\[z_1(x,y) = x^2 + .5y^2\]

- **AXES** — Shows standard xyz axes.
- **BOX** — Shows 3-dimensional box axes.

The edges of the box are determined by the Window variables \[\text{xmin}, \text{ xmax}\], etc.

In many cases, the origin \((0,0,0)\) is inside the box, not at a corner. For example, if \[\text{xmin} = \text{ymin} = \text{zmin} = -10\] and \[\text{xmax} = \text{ymax} = \text{zmax} = 10\], the origin is at the center of the box.
**Examples of Style Settings**

**Note:** WIRE FRAME is faster to graph and may be more convenient when you’re experimenting with different shapes.

To display the valid Style settings, highlight the current setting and press ▇.

- **WIRE FRAME** — Shows the 3D shape as a transparent wire frame.
- **HIDDEN SURFACES** — Uses shading to differentiate the two sides of the 3D shape.

Later sections in this module describe CONTOUR LEVELS, WIRE AND CONTOUR, and implicit plots.

**Be Aware of Possible Optical Illusions**

The eye angles used to view a graph (eyeθ, eyeφ, and eyeψ Window variables) can result in optical illusions that cause you to lose perspective on a graph. Typically, most optical illusions occur when the eye angles are in a negative quadrant of the coordinate system.
Optical illusions may be more noticeable with box axes. For example, it may not be immediately obvious which is the “front” of the box.

<table>
<thead>
<tr>
<th>Looking down from above the xy plane</th>
<th>Looking up from below the xy plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Looking down diagram]</td>
<td>![Looking up diagram]</td>
</tr>
</tbody>
</table>

\[ \text{eye}\theta = 20, \text{eye}\phi = 55, \text{eye}\psi = 0 \quad \text{eye}\theta = 20, \text{eye}\phi = 120, \text{eye}\psi = 0 \]

**Note:** The first two examples show the graphs as displayed on the screen. The second two examples use artificial shading (which is not displayed on the screen) to show the front of the box.

To minimize the effect of optical illusions, use the GRAPH FORMATS dialog box to set Style = HIDDEN SURFACE.

**Contour Plots**

In a contour plot, a line is drawn to connect adjacent points on the 3D graph that have the same z value. This module discusses the CONTOUR LEVELS and WIRE AND CONTOUR graph format styles.
Selecting the Graph Format Style

In 3D graphing mode, define an equation and graph it as you would any 3D equation, with the following exception. Display the GRAPH FORMATS dialog box by pressing \( \text{F1 9} \) from the Y= Editor, Window editor, or Graph screen. Then set:

Style = CONTOUR LEVELS
– or –
Style = WIRE AND CONTOUR

• For CONTOUR LEVELS, only the contours are shown.
  - The viewing angle is set initially so that you are viewing the contours by looking down the z axis. You can change the viewing angle as necessary.
  - The graph is shown in expanded view. To switch between expanded and normal view, press \( \text{X} \).
  - The Labels format is set to OFF automatically.

• For WIRE AND CONTOUR, the contours are drawn on a wire frame view. The viewing angle, view (expanded or normal), and Labels format retain their previous settings.

Notes:

• From the Graph screen, you can switch from one graph format style to the next (skipping IMPLICIT PLOT) by pressing: \( \text{I} \)

• Pressing:
  \( \text{I} \)
to select CONTOUR LEVELS does not affect the viewing angle, view, or Labels
format as it does if you use:

<table>
<thead>
<tr>
<th>Style</th>
<th>( z_1(x,y) = \frac{x^3 y - y^3 x}{390} )</th>
<th>( z_1(x,y) = x^2 + 0.5y^2 - 5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTOUR LEVELS</td>
<td>Looking down z axis</td>
<td></td>
</tr>
</tbody>
</table>

Using \( \text{eye}\theta = 20, \text{eye}\phi = 70, \text{eye}\psi = 0 \)

**Note:** These examples use the same \( x, y, \) and \( z \) Window variable values as a **ZoomStd** viewing cube. If you use **ZoomStd**, press \( Z \) to look down the \( z \) axis. Do not confuse the contours with the grid lines. The contours are darker.
How Are Z Values Determined?

You can set the ncontour Window variable ([WINDOW]) to specify the number of contours that will be evenly distributed along the displayed range of z values, where:

\[
\text{increment} = \frac{z_{\text{max}} - z_{\text{min}}}{\text{ncontour} + 1}
\]

The z values for the contours are:

\[
\begin{align*}
&z_{\text{min}} + \text{increment} \\
&z_{\text{min}} + 2(\text{increment}) \\
&z_{\text{min}} + 3(\text{increment}) \\
&\vdots \\
&z_{\text{min}} + \text{ncontour}(\text{increment})
\end{align*}
\]

If ncontour=5 and you use the standard viewing window (zmin=-10 and zmax=10), the increment is 3.333. Five contours are drawn for z=-6.666, -3.333, 0, 3.333, and 6.666.

Note, however, that a contour is not drawn for a z value if the 3D graph is not defined at that z value.
**(Drawing a Contour for the Z Value of a Selected Point Interactively)**

If a contour graph is currently displayed, you can specify a point on the graph and draw a contour for the corresponding z value.

1. **To display the Draw menu, press:**
   
   ![2nd [F6]

2. **Select 7:Draw Contour.**

3. **Either:**
   - Type the point’s x value and press **[ENTER]**, and then type the y value and press **[ENTER]**.
   - or –
   - Move the cursor to the applicable point. (The cursor moves along the grid lines.) Then press **[ENTER]**.

For example, suppose the current graph is \( z_1(x,y) = x^2 + 0.5y^2 - 5 \). If you specify \( x=2 \) and \( y=3 \), a contour is drawn for \( z=3.5 \).

**Note:** Any existing contours remain on the graph. To remove the default contours, display the Window editor (\( [\text{WINDOW}] \)) and set \( \text{ncontour}=0 \).
**Drawing Contours for Specified Z Values**

From the Graph screen, display the Draw menu and then select `8:DrwCtour`. The Home screen is displayed automatically with `DrwCtour` in the entry line. You can then specify one or more z values individually or generate a sequence of z values.

Some examples are:

<table>
<thead>
<tr>
<th>DrwCtour</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DrwCtour 5</strong></td>
<td>Draws a contour for z=5.</td>
</tr>
<tr>
<td><strong>DrwCtour {1,2,3}</strong></td>
<td>Draws contours for z=1, 2, and 3.</td>
</tr>
<tr>
<td><strong>DrwCtour seq(n,n,-10,10,2)</strong></td>
<td>Draws contours for a sequence of z values from -10 through 10 in steps of 2 (-10, -8, -6, etc.).</td>
</tr>
</tbody>
</table>

**Note:** To remove the default contours, use `F2 [WINDOW]` and set `ncontour=0`.

The specified contours are drawn on the current 3D graph. (A contour is not drawn if the specified z value is outside the viewing cube or if the 3D graph is not defined at that z value.)

**Notes about Contour Plots**

For a contour plot:

- You can use the cursor keys to animate the contour plot.
- You cannot trace (`F3`) the contours themselves. However, you can trace the wire frame as seen when Style=WIRE AND CONTOUR.
- It may take awhile to evaluate the equation initially.
• Because of possible long evaluation times, you first may want to experiment with your 3D equation by using Style=WIRE FRAME. The evaluation time is much shorter. Then, after you’re sure you have the correct Window variable values, display the Graph Formats dialog box and set Style=CONTOUR LEVELS or WIRE AND CONTOUR.

Example: Contours of a Complex Modulus Surface

The complex modulus surface given by \( z(a,b) = \text{abs}(f(a+bi)) \) shows all the complex zeros of any polynomial \( y=f(x) \).

Example

In this example, let \( f(x)=x^3+1 \). By substituting the general complex form \( x+y\cdot i \) for \( x \), you can express the complex surface equation as \( z(x,y)=\text{abs}((x+y\cdot i)^3+1) \).

1. Use [MODE] to set Graph=3D.
2. Press [Y=], and define the equation:
   \[
z_1(x,y)=\text{abs}((x+y\cdot i)^3+1)
   \]
3. Press [WINDOW], and set the Window variables as shown.
4. Display the Graph Formats dialog box:  
   [Submit] Turn on the axes, set  
   **Style = CONTOUR LEVELS**, and return to  
   the Window editor.

5. Press [GRAPH] to graph the equation.

   It will take awhile to evaluate the graph; so be patient. When the graph is displayed,  
   the complex modulus surface touches the xy plane at exactly the complex zeros of  
   the polynomial:

   \[-1, \frac{1}{2} + \frac{\sqrt{3}}{2} i, \text{ and } \frac{1}{2} - \frac{\sqrt{3}}{2} i\]

6. Press [F3], and move the trace cursor to  
   the zero in the fourth quadrant.

   The coordinates let you estimate  
   \(0.428 - 0.857i\) as the zero.

7. Press [ESC]. Then use the cursor keys to  
   animate the graph and view it from  
   different eye angles.

   This example shows  
   \(\text{eye}_\theta = 70, \text{ eye}_\phi = 70, \text{ and } \text{eye}_\psi = 0\).
Notes:

- For more accurate estimates, increase the `xgrid` and `ygrid` Window variables. However, this increases the graph evaluation time.

- When you animate the graph, the screen changes to normal view. Use □ to toggle between normal and expanded views.

**Implicit Plots**

An implicit plot is used primarily as a way to graph 2D implicit forms that cannot be graphed in function graphing mode. Technically, an implicit plot is a 3D contour plot with a single contour drawn for $z=0$ only.

**Explicit and Implicit Forms**

In 2D function graphing mode, equations have an explicit form $y=f(x)$, where $y$ is unique for each value of $x$.

Many equations, however, have an implicit form $f(x,y)=g(x,y)$, where you cannot explicitly solve for $y$ in terms of $x$ or for $x$ in terms of $y$. $y$ is not unique for each $x$, so you cannot graph this in function graphing mode.
By using implicit plots in 3D graphing mode, you can graph these implicit forms without solving for y or x.

Rearrange the implicit form as an equation set to zero.

\[ f(x,y) - g(x,y) = 0 \]

In the Y= Editor, enter the non-zero side of the equation. This is valid because an implicit plot automatically sets the equation equal to zero.

For example, given the ellipse equation shown to the right, enter the implicit form in the Y= Editor.

\[ z_1(x,y) = f(x,y) - g(x,y) \]

\[ \text{If } x^2 + 0.5y^2 = 30, \text{ then } z_1(x,y) = x^2 + 0.5y^2 - 30. \]

**Notes:** You can also graph many implicit forms if you either:

- Express them as parametric equations.
- Break them into separate, explicit functions.

**Selecting the Graph Format Style**

In 3D graphing mode, define an appropriate equation and graph it as you would any 3D equation, with the following exception. Display the GRAPH FORMATS dialog box from the Y= Editor, Window editor, or Graph screen:
Note: From the Graph screen, you can switch to the other graph format styles by pressing: 

However, to return to IMPLICIT PLOT press:

• The viewing angle is set initially so that you are viewing the plot by looking down the z axis. You can change the viewing angle as necessary.
• The plot is shown in expanded view. To switch between expanded and normal view, press .
• The Labels format is set to OFF automatically.

<table>
<thead>
<tr>
<th>Style</th>
<th>( x^2 - y^2 = 4 )</th>
<th>( \sin(x) + \cos(y) = e(x \cdot y) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPLICIT PLOT</td>
<td>( z_1(x,y) = x^2 - y^2 - 4 )</td>
<td>( z_1(x,y) = \sin(x) + \cos(y) - e(x \cdot y) )</td>
</tr>
</tbody>
</table>
Note: These examples use the same x, y, and z Window variable values as a ZoomStd viewing cube. If you use ZoomStd, press Z to look down the z axis.

Notes About Implicit Plots

For an implicit plot:

• The ncontour Window variable has no affect. Only the z=0 contour is drawn, regardless of the value of ncontour. The displayed plot shows where the implicit form intersects the xy plane.

• You can use the cursor keys to animate the plot.

• You cannot trace (F3) the implicit plot itself. However, you can trace the unseen wire frame graph of the 3D equation.

• It may take awhile to evaluate the equation initially.

• Because of possible long evaluation times, you first may want to experiment with your 3D equation by using Style=WIRE FRAME. The evaluation time is much shorter. Then, after you’re sure you have the correct Window variable values, set Style=IMPLICIT PLOT.

Example: Implicit Plot of a More Complicated Equation

You can use the IMPLICIT PLOT graph format style to plot and animate a complicated equation that cannot be graphed otherwise. Although it may take a long time to evaluate such a graph, the visual results can justify the time required.
Graph the equation $\sin(x^4 + y - x^3y) = .1$.

1. Use $\text{MODE}$ to set $\text{Graph}=3D$.

2. Press $\downarrow [Y=]$, and define the equation:
   
   $$z_1(x,y) = \sin(x^4 + y - x^3y) - .1$$

3. Press $\downarrow [\text{WINDOW}]$, and set the Window variables as shown.

4. Press:
   
   $\downarrow \downarrow \text{Turn on the axes, set}$
   
   $\textbf{Style} = \text{IMPLICIT PLOT}$, and return to the Window editor.

5. Press $\downarrow [\text{GRAPH}]$ to graph the equation.
   
   It will take awhile to evaluate the graph; so be patient.

The graph shows where $\sin(x^4 + y - x^3y) = .1$.
6. Use the cursor keys to animate the graph and view it from different eye angles.

**Note:** For more detail, increase the `xgrid` and `ygrid` Window variables. However, this increases the graph evaluation time. In expanded view, this example shows `eyeθ=−127.85`, `eyeφ=52.86`, and `eyeψ=−18.26`.

**Note:** When you animate the graph, the screen changes to normal view. Press □ to switch between normal and expanded views.
Differential Equation Graphing

Overview of Steps in Graphing Differential Equations

To graph differential equations, use the same general steps used for $y(x)$ functions as described in *Basic Function Graphing*. Any differences are described on the following pages.

Graphing Differential Equations

1. Set **Graph** mode (MODE) to **DIFF EQUATIONS**. Also set **Angle** mode, if necessary.

2. Define equations and, optionally, initial conditions on Y= Editor ([Y=]).

3. Select ([F4]) which defined functions to graph.

   **Note**: To turn off any stat data plots, press [F5] 5 or use [F4] to deselect them.
4. Set the display style for a function.
   \[ \text{Display} \begin{cases} 2nd [F6] \end{cases} \]

5. Set the graph format. **Solution Method** and **Fields** are unique to differential equations.
   \[ \text{F1} 9 \]
   \[ \text{or} \]
   \[ \text{Display} \begin{cases} \downarrow \uparrow \end{cases} \]

   **Note:** The **Fields** format is critical, depending on the order of the equation.

6. Set the axes as applicable, depending on the Fields format.
   \[ \text{Display} \begin{cases} 2nd [F7] \end{cases} \]

   **Note:** Valid **Axes** settings depend on the Fields format.

7. Define the viewing window (\[ [\text{WINDOW}] \]).

   **Note:** Depending on the **Solution Method** and **Fields** formats, different **Window** variables are displayed. \[ \text{F2} \text{ Zoom} \] also changes the viewing window.

\[ \begin{align*}
t_0 &= 9, \\
t_{\text{max}} &= 10, \\
t_{\text{step}} &= 1, \\
t_{\text{plot}} &= 0, \\
x_{\text{min}} &= -10, \\
x_{\text{max}} &= 110, \\
x_{\text{scale}} &= 10, \\
y_{\text{min}} &= -10, \\
y_{\text{max}} &= 120, \\
y_{\text{scale}} &= 10, \\
\text{ncurves} &= 0, \\
giftol &= .001 \\
fldres &= 20. \end{align*} \]
Differences in Diff Equations and Function Graphing

This module assumes that you already know how to graph $y(x)$ functions as described in Basic Function Graphing. This section describes the differences.

Setting the Graph Mode

Use `MODE` to set Graph = DIFF EQUATIONS before you define differential equations or set Window variables. The Y= Editor and the Window Editor let you enter information for the current Graph mode setting only.

Defining Differential Equations on the Y= Editor

Use $t_0$ to specify when initial conditions occur. You can also set $t_0$ in the Window Editor. Use $y_i$ to specify one or more initial conditions for the corresponding differential equation. You can define differential equations $y_1'(t)$ through $y_{99}'(t)$. 

8. Graph the selected functions (`[GRAPH]`).
**Note:** You can use the **Define** command from the Home screen to define functions and equations.

When entering equations in the Y= Editor, do not use \( y(t) \) formats to refer to results. For example:

\[
\text{Enter: } y_1' = 0.001y_1(100-y_1) \\
\text{Not: } y_1' = 0.001y_1(t)(100-y_1(t))
\]

Only 1st-order equations can be entered in the Y= Editor. To graph 2nd- or higher-order equations, you must enter them as a system of 1st-order equations.

Detailed information is available on setting initial conditions.

**Selecting Differential Equations**

You can use \[ \text{F4} \] to select a differential equation, but not its initial condition.

**Important:** Selecting \( y_1' \) will graph the \( y_1 \) solution curve, not the derivative \( y_1' \), depending on the axis setting.
Selecting the Display Style

With the Style menu, only the **Line**, **Dot**, **Square**, **Thick**, **Animate**, and **Path** styles are available. **Dot** and **Square** mark only those discrete values (in **tstep** increments) at which a differential equation is plotted.

2nd [F6]

Setting Graph Formats

From the Y= Editor, Window Editor, or Graph screen, press:

F1 9
— or —

The formats affected by differential equations are:

<table>
<thead>
<tr>
<th>Graph format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph Order</td>
<td>Not available.</td>
</tr>
<tr>
<td>Solution Method</td>
<td>Specifies the method used to solve the differential equations.</td>
</tr>
<tr>
<td></td>
<td>• RK — Runge-Kutta method. For information about the algorithm used for this method, refer to the <em>Technical Reference</em> module.</td>
</tr>
<tr>
<td></td>
<td>• EULER — Euler method.</td>
</tr>
<tr>
<td></td>
<td>• The method lets you choose either greater accuracy or speed. Typically, RK is more accurate than EULER but takes longer to find the solution.</td>
</tr>
</tbody>
</table>
### Graph format

<table>
<thead>
<tr>
<th>Graph format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields</td>
<td>Specifies whether to draw a field for the differential equation.</td>
</tr>
<tr>
<td></td>
<td>• SLPFLD — Draws a slope field for only one 1st-order equation, with t on the x axis and the solution on the y axis.</td>
</tr>
<tr>
<td></td>
<td>• DIRFLD — Draws a direction field for only one 2nd-order equation (or system of two 1st-order equations), with axes determined by the custom axes settings.</td>
</tr>
<tr>
<td></td>
<td>• FLDOFF — Does not display a field. This is valid for equations of any order, but you must use it for 3rd- or higher-order. You must enter the same number of initial conditions for all equations in the Y= Editor.</td>
</tr>
</tbody>
</table>

**Important:** The Fields graph format is critical in successfully graphing differential equations.

**Note:** If you press `[ENTER]` while a slope or direction field is being drawn, the graph pauses after the field is drawn but before the solutions are plotted. Press `[ENTER]` again to continue. To cancel graphing, press `[ON]`.

### Setting Axes

In the Y= Editor, **Axes** may or may not be available, depending on the current graph format.

If it is available, you can select the axes that are used to graph the differential equations. 

**Axes Selection**

1. Use `[2ND] [F7]` to access the axes menu.
2. Select the desired axes for graphing.
Differential equation graphs use the following Window variables. Depending on the Solution Method and Fields graph formats, not all of these variables are listed in the Window Editor ([WINDOW]) at the same time.

### Axes Description

<table>
<thead>
<tr>
<th>Axes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>Plots t on the x axis and y (the solutions to the selected differential equations) on the y axis.</td>
</tr>
<tr>
<td>CUSTOM</td>
<td>Lets you select the x and y axes.</td>
</tr>
</tbody>
</table>

### Window Variables

Differential equation graphs use the following Window variables. Depending on the Solution Method and Fields graph formats, not all of these variables are listed in the Window Editor ([WINDOW]) at the same time.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t0</td>
<td>Time at which the initial conditions entered in the Y= Editor occur. You can set t0 in the Window Editor and Y= Editor. (If you set t0 in the Y= Editor, tplot is set to the same value automatically.)</td>
</tr>
<tr>
<td>tmax, tstep</td>
<td>Used to determine the t values where the equations are plotted: y'(t0) y'(t0+tstep) y'(t0+2*tstep) ... not to exceed ... y'(tmax) If Fields = SLPFLD, tmax is ignored. Equations are plotted from t0 to both edges of the screen in tstep increments.</td>
</tr>
</tbody>
</table>
### Variable Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tplot</strong></td>
<td>First t value plotted. If this is not a <strong>tstep</strong> increment, plotting begins at the next <strong>tstep</strong> increment. In some situations, the first points evaluated and plotted starting at <strong>t0</strong> may not be interesting visually. By setting <strong>tplot</strong> greater than <strong>t0</strong>, you can start the plot at the interesting area, which speeds up the graphing time and avoids unnecessary clutter on the Graph screen.</td>
</tr>
<tr>
<td><strong>tmax</strong></td>
<td>If <strong>tmax &lt; t0</strong>, <strong>tstep</strong> must be negative. If <strong>Fields=SLPFLD</strong>, <strong>tplot</strong> is ignored and is assumed to be the same as <strong>t0</strong>.</td>
</tr>
</tbody>
</table>

**Note:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>xmin, xmax, ymin, ymax</strong></td>
<td>Boundaries of the viewing window.</td>
</tr>
<tr>
<td><strong>xscl, yscl</strong></td>
<td>Distance between tick marks on the x and y axes.</td>
</tr>
<tr>
<td><strong>ncurves</strong></td>
<td>Number of solution curves (0 through 10) that will be drawn automatically if you do not specify an initial condition. By default, <strong>ncurves = 0</strong>.</td>
</tr>
</tbody>
</table>
When \texttt{ncurves} is used, \texttt{t0} is set temporarily at the middle of the screen and initial conditions are distributed evenly along the y axis, where:

\[
\text{increment} = \frac{\text{ymax} - \text{ymin}}{\text{ncurves} + 1}
\]

The y values for the initial conditions are:
\begin{align*}
\text{ymin} + \text{increment} \\
\text{ymin} + 2\times(\text{increment}) \\
\vdots \\
\text{ymin} + \text{ncurves}\times(\text{increment})
\end{align*}

\begin{tabular}{|l|l|}
\hline
\textbf{Variable} & \textbf{Description} \\
\hline
diftol & (Solution Method = RK only) Tolerance used by the RK method to help select a step size for solving the equation; must be \(\geq 1\times10^{-14}\). \\
\hline
fldres & (Fields = SLPFLD or DIRFLD only) Number of columns (1 through 80) used to draw a slope or direction field across the full width of the screen. \\
\hline
Estep & (Solution Method = EULER only) Euler iterations between \texttt{tstep} values; must be an integer >0. For more accuracy, you can increase \texttt{Estep} without plotting additional points. \\
\hline
dtime & (Fields = DIRFLD only) Point in time at which a direction field is drawn. \\
\hline
\end{tabular}
Standard values (set when you select 6:ZoomStd from the F2 Zoom toolbar menu) are:

\begin{align*}
  t0 &= 0. \\
  tmax &= 10. \\
  tstep &= .1 \\
  tplot &= 0. \\
  xmin &= -1. \\
  xmax &= 10. \\
  xscl &= 1. \\
  ymin &= -10. \\
  ymax &= 10. \\
  yscl &= 1. \\
  ncurves &= 0. \\
  diftol &= .001 \\
  Estep &= 1. \\
  fldres &= 14. \\
  dtime &= 0.
\end{align*}

You may need to change the standard values for the t variables to ensure that sufficient points are plotted.

**The fldpic System Variable**

When a slope or direction field is drawn, a picture of the field is stored automatically to a system variable named fldpic. If you perform an operation that regraphs the plotted equations but does not affect the field, the TI-89 Titanium reuses the picture in fldpic instead of having to redraw the field. This can speed up the regraphing time significantly.

fldpic is deleted automatically when you exit the differential equation graphing mode or when you display a graph with Fields = FLDOFF.

**Exploring a Graph**

As in function graphing, you can explore a graph by using the following tools. Any displayed coordinates are shown in rectangular or polar form as set in the graph format.

<table>
<thead>
<tr>
<th>Tool</th>
<th>For Differential Equation Graphs:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free-Moving Cursor</td>
<td>Works just as it does for function graphs.</td>
</tr>
</tbody>
</table>
### Tool: For Differential Equation Graphs:

<table>
<thead>
<tr>
<th>Tool</th>
<th>For Differential Equation Graphs:</th>
</tr>
</thead>
</table>
| **F2** Zoom | Works just as it does for function graphs.  
- Only \( x (\text{xmin}, \text{xmax}, \text{xscl}) \) and \( y (\text{ymin}, \text{ymax}, \text{yscl}) \) Window variables are affected.  
- The \( t \) Window variables \((\text{t0}, \text{tmax}, \text{tstep}, \text{tplot})\) are not affected unless you select **6:ZoomStd** (which sets all Window variables to their standard values). |
| **F3** Trace | Lets you move the cursor along the curve one \( \text{tstep} \) at a time. To move approximately ten plotted points at a time, press **2nd** \( \square \) or **2nd** \( \rightarrow \). \nIf you enter initial conditions in the Y= Editor or let the \text{ncurves} Window variable plot curves automatically, you can trace the curves. If you use: \n\begin{align*} \text{IC} & \quad \text{2nd} \quad \text{[F8]} \\
\text{IC from the Graph screen to select initial conditions interactively, you cannot trace the curves.} \\
\text{QuickCenter applies to all directions. If you move the cursor off the screen (top or bottom, left or right), press} \quad \text{[ENTER]} \quad \text{to center the viewing window on the cursor location.} \\
\text{Use} \quad \square \quad \text{or} \quad \rightarrow \quad \text{to view results on all plotted curves.} \end{align*} |
| **F5** Math | Only **1:Value** is available.  
- With TIME axes, the \( y(t) \) solution value (represented by \( yc \)) is displayed for a specified \( t \) value.  
- With CUSTOM axes, the values that correspond to \( x \) and \( y \) depend on the axes you choose. |

**Note:** During a trace, you can move the cursor to a particular point by typing a value for \( t \) and pressing **[ENTER]**. You can use QuickCenter at any time during a trace, even if the cursor is still on the screen.
Setting the Initial Conditions

You can enter initial conditions in the Y= Editor, let the TI-89 Titanium calculate initial conditions automatically, or select them interactively from the Graph screen.

**Entering Initial Conditions in the Y= Editor**

You can specify one or more initial conditions in the Y= Editor. To specify more than one, enter them as a list enclosed in braces { } and separated by commas.

To enter initial conditions for the \( y_1' \) equation, use the \( y_1 \) line, etc.

To specify when the initial conditions occur, use \( t_0 \). This is also the first \( t \) evaluated for the graph.

To graph a family of solutions, enter a list of initial conditions.

For a 2nd- or higher-order differential equation, you must define a system of 1st-order equations in the Y= Editor.

If you enter initial conditions, you must enter the same number of initial conditions for each equation in the system. Otherwise, a Dimension error occurs.
If You Do Not Enter an Initial Condition in the Y= Editor

If you do not enter initial conditions, the ncurves Window variable ([WINDOW]) specifies the number of solution curves graphed automatically. By default, ncurves = 0. You can enter a value from 0 through 10. However, the Fields graph format and the Axes setting determine whether ncurves is used.

<table>
<thead>
<tr>
<th>If Fields =</th>
<th>Then:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLPFLD</td>
<td>Uses ncurves, if not set to 0, to graph curves.</td>
</tr>
<tr>
<td>DIRFLD</td>
<td>Ignores ncurves. Does not graph any curves.</td>
</tr>
<tr>
<td>FLDOFF</td>
<td>Uses ncurves if Axes = TIME (or if Axes = Custom and the x axis is t). Otherwise, a Diff Eq setup error occurs.</td>
</tr>
</tbody>
</table>

When ncurves is used, t0 is set temporarily at the middle of the Graph screen. However, the value of t0 as set in the Y= Editor or Window Editor is not changed.

Notes:

• Without entering initial conditions, use SLPFLD (with ncurves=0) or DIRFLD to display a slope or direction field only.

• SLPFLD is for a single 1st-order equation only. DIRFLD is for a 2nd-order equation (or system of two 1st-order equations) only.
Selecting an Initial Condition Interactively from the Graph Screen

When a differential equation is graphed (regardless of whether a solution curve is displayed), you can select a point on the Graph screen and use it as an initial condition.

<table>
<thead>
<tr>
<th>If Fields =</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLPFLD</td>
<td>Press:</td>
</tr>
<tr>
<td>– or –</td>
<td>2nd [F8]</td>
</tr>
<tr>
<td>DIRFLD</td>
<td>Specify an initial condition. Either:</td>
</tr>
<tr>
<td></td>
<td>• Move the cursor to the applicable point and press [ENTER]. – or –</td>
</tr>
<tr>
<td></td>
<td>• For each of the two coordinates, type a value and press [ENTER].</td>
</tr>
<tr>
<td></td>
<td>- For SLPFLD (1st-order only), enter values for ( t_0 ) and ( y(t_0) ).</td>
</tr>
<tr>
<td></td>
<td>- For DIRFLD (2nd-order or system of two 1st-order equations only), enter values for both ( y(t_0) ) initial conditions, where ( t_0 ) is the value set in the Y= Editor or Window Editor.</td>
</tr>
</tbody>
</table>

A circle marks the initial condition and the solution curve is drawn.
**Note about Tracing a Solution Curve**

When you enter initial conditions in the Y= Editor or let `ncurves` graph solution curves automatically, you can use [F3] to trace the curves. However, you cannot trace a curve drawn by selecting an initial condition interactively. These curves are drawn, not plotted.
Defining a System for Higher-Order Equations

In the Y= Editor, you must enter all differential equations as 1st-order equations. If you have an nth-order equation, you must transform it into a system of n 1st-order equations.

Transforming an Equation into a 1st-Order System

A system of equations can be defined in various ways, but the following is a general method.

1. Rewrite the original differential equation as necessary.
   
   a) Solve for the highest-ordered derivative.

   \[
   y'' + y' + y = e^x
   \]
   
   \[
   y'' = e^x - y' - y
   \]
   
   b) Express it in terms of y and t.

   \[
   y'' = e^t - y' - y
   \]
   
   c) On the right side of the equation only, substitute to eliminate any references to derivative values.

   **Note:** To produce a 1st-order equation, the right side must contain non-derivative variables only.
<table>
<thead>
<tr>
<th>In place of:</th>
<th>Substitute:</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>y1</td>
</tr>
<tr>
<td>y'</td>
<td>y2</td>
</tr>
<tr>
<td>y''</td>
<td>y3</td>
</tr>
<tr>
<td>y'''</td>
<td>y4</td>
</tr>
<tr>
<td>y(4)</td>
<td>y5</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

\[ y'' = e^t - y2 - y1 \]

Do not substitute on the left side at this time.

d) On the left side of the equation, substitute for the derivative value as shown below.

<table>
<thead>
<tr>
<th>In place of:</th>
<th>Substitute:</th>
</tr>
</thead>
<tbody>
<tr>
<td>y'</td>
<td>y1'</td>
</tr>
<tr>
<td>y''</td>
<td>y2'</td>
</tr>
<tr>
<td>y'''</td>
<td>y3'</td>
</tr>
<tr>
<td>y(4)</td>
<td>y4'</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

\[ y2' = e^t - y2 - y1 \]
2. On the applicable lines in the Y= Editor, define the system of equations as:

\[
\begin{align*}
y_1' &= y_2 \\
y_2' &= y_3 \\
y_3' &= y_4 \\
\vdots \\
y_n' &= \text{your nth-order equation}
\end{align*}
\]

**Note:** Based on the above substitutions, the y' lines in the Y= Editor represent:

\[
\begin{align*}
y_1' &= y' \\
y_2' &= y'' \\
\vdots
\end{align*}
\]

Therefore, this example’s 2nd-order equation is entered on the y2' line.

In a system such as this, the solution to the y1' equation is the solution to the nth-order equation. You may want to deselect any other equations in the system.

**Example of a 2nd-Order Equation**

The 2nd-order differential equation \(y'' + y = 0\) represents a simple harmonic oscillator. Transform this into a system of equations for the Y= Editor. Then, graph the solution for initial conditions \(y(0) = 0\) and \(y'(0) = 1\).
Example

1. Press [MODE] and set **Graph=DIFF EQUATIONS**.

2. Define a system of equations for the 2nd-order equation.

   Rewrite the equation and make the necessary substitutions.

   \[
   y'' + y = 0 \\
   y'' = -y \\
   y'' = -y_1 \\
   y_2' = -y_1
   \]

3. In the Y= Editor ([Y=]), enter the system of equations.

4. Enter the initial conditions:

   \( y_{i1} = 0 \) and \( y_{i2} = 1 \)

   **Note**: \( t_0 \) is the time at which the initial conditions occur. It is also the first \( t \) evaluated for the graph. By default, \( t_0 = 0 \).

5. Press:

   - \( \text{F1 9} \)
   - \( \text{F2} \) and set **Axes=ON**, **Labels=OFF**, **Solution Method=RK**, and **Fields=DIRFLD**.

   **Important**: For 2nd-order equations, you must set **Fields=DIRFLD** or **FLDOFF**.
6. In the Y= Editor, press:

\[2^{nd}\ [F7]\] and make sure

**Axes = CUSTOM** with \(y1\) and \(y2\) as the axes.

**Important:** **Fields=DIRFLD** cannot plot a time axis. An **Invalid Axes** error occurs if **Axes=TIME** or if \(t\) is set as a **CUSTOM** axis.

7. In the Window Editor ([WINDOW]), set the Window variables.

\[
t0=0 \quad x_{\min}=-2 \quad n_{\text{curves}}=0
\]
\[
t_{\max}=10 \quad x_{\max}=2 \quad d_{\text{ftol}}=.001
\]
\[
t_{\text{step}}=.1 \quad x_{\text{scl}}=1 \quad f_{\text{ldres}}=14
\]
\[
t_{\text{plot}}=0 \quad y_{\min}=-2 \quad d_{\text{time}}=0
\]
\[
y_{\max}=2 \quad y_{\text{scl}}=1
\]

8. Display the Graph screen ([GRAPH]).

\[x\text{ axis }= y1 = y\]
\[y\text{ axis }= y2 = y'\]

If you select **ZoomSqr** ([F2] 5), you can see that the phase-plane orbit is actually a circle. However, **ZoomSqr** will change your Window variables.
To examine this harmonic oscillator in more detail, use a split screen to graph the manner in which \( y \) and \( y' \) change with respect to time (t).

9. Press \( \text{MODE} \) and change the mode settings on \textbf{Page 2} as shown. Then close the \textbf{MODE} dialog box, which redraws the graph.

\textbf{Note:} To display different graphs in both parts of a split screen, you must use the \textbf{2-graph} mode.

10. Press \( \text{2nd} \ [\rightarrow] \) to switch to the right side of the split screen.

11. Use \( \text{F4} \) to select \( y_1' \) and \( y_2' \).

The right side uses the same equations as the left side. However, no equations are selected initially in the right side.

12. Press:

\( \text{F1} \ 9 \)

— or —

\( \text{Set Fields = FLDOFF.} \)

\textbf{Important:} Because \textbf{Fields=DIRFLD} cannot plot a time axis, you must change the Fields setting. \textbf{FLDOFF} turns off all fields.

13. In the \textbf{Y= Editor}, press:

\( \text{2nd} \ [\text{F7}] \) and make sure \textbf{Axes = TIME}. 

Example of a 3rd-Order Equation

For the 3rd-order differential equation $y''' + 2y'' + 2y' + y = \sin(x)$, write a system of equations to enter in the Y= Editor. Then graph the solution as a function of time. Use initial conditions $y(0) = 0$, $y'(0) = 1$, and $y''(0) = 1$. 

14. In the Window Editor, change $y_{\text{min}}$ and $y_{\text{max}}$ as shown to the right.

   \textbf{Note:} When you enter 2-graph mode, Window variables for the right side are set to their defaults.

15. Press $\bullet \ [\text{GRAPH}]$ to display the Graph screen for graph #2.

   The left side shows the phase-plane orbit. The right side shows the solution curve and its derivative.

16. To return to a full screen of the original graph, press \texttt{2nd} \ [\text{VAR}] to switch to the left side. Then press \texttt{MODE} and change the \texttt{Split Screen} setting.

\texttt{Split Screen = FULL}
1. Press **MODE** and set **Graph=DIFF EQUATIONS**.

2. Define a system of equations for the 3rd-order equation.

   Rewrite the equation and make the necessary substitutions.

   \[ y''' + 2y'' + 2y' + y = \sin(x) \]
   \[ y''' = \sin(x) - 2y'' - 2y' - y \]
   \[ y''' = \sin(t) - 2y'' - 2y' - y \]
   \[ y''' = \sin(t) - 2y3 - 2y2 - y1 \]
   \[ y3' = \sin(t) - 2y3 - 2y2 - y1 \]

3. In the Y= Editor (\[ \text{Y=} \]), enter the system of equations.

4. Enter the initial conditions:
   \[ yi1=0, \ yi2=1, \text{ and } yi3=1 \]
   **Note**: \( t0 \) is the time at which the initial conditions occur. By default, \( t0=0 \).

5. Be sure that only \( y1' \) is selected. Use \( \text{F4} \) to deselect any other equations.

   **Important**: The solution to the \( y1' \) equation is the solution to the 3rd-order equation.
6. Press: \( \text{F1} \ 9 \)
   — or —
   Set Axes = ON, Labels = ON, Solution Method = RK, and Fields = FLDOFF.

   **Important:** For 3rd- or higher-order equations, you must set Fields=FLDOFF. Otherwise, an **Undefined variable** error occurs when graphing.

7. In the Y= Editor, press:
   \( \text{Graph} \ \text{F7} \) Set Axes = TIME.

   **Note:** With Axes=TIME, the solution to the selected equation is plotted against time (t).

8. In the Window Editor (\( \text{Graph} \ \text{F3} \ [\text{Window}] \)), set the Window variables.

   \[
   \begin{align*}
t0 &= 0 \\
tmax &= 10 \\
tstep &= .1 \\
tplot &= 0 \\
xmin &= -1 \\
xmax &= 10 \\
xscl &= 1. \\
ymin &= -3 \\
ymax &= 3 \\
y scl &= 1
   \end{align*}
   \]

   \[
   \begin{align*}
ncurves &= 0 \\
diftol &= .001
   \end{align*}
   \]
9. Display the Graph screen ([GRAPH]).

**Note:** To find the solution at a particular time, use \[F3\] to trace the graph.

**Setting Axes for Time or Custom Plots**

Setting the axes can give you great flexibility in graphing differential equations. Custom axes are particularly effective for showing different kinds of relationships.

**Displaying the AXES Dialog Box**

From the Y= Editor, press:

\[2nd \, [F7]\]

If \textbf{Fields = SLPFLD}, \textbf{Axes} is unavailable.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
</table>
| Axes  | TIME — Plots \(t\) on the \(x\) axis and \(y\) (solutions to all selected differential equations) on the \(y\) axis.  
CUSTOM — Lets you select the \(x\) and \(y\) axes. |
Using the predator-prey model from biology, determine the numbers of rabbits and foxes that maintain population equilibrium in a certain region. Graph the solution using both time and custom axes.
Predator-Prey Model

Use the two coupled 1st-order differential equations:

\[
y_1' = -y_1 + 0.1y_1 \cdot y_2 \quad \text{and} \quad y_2' = 3y_2 - y_1 \cdot y_2
\]

where:

\[
y_1 = \text{Population of foxes}
\]

\[
y_{i1} = \text{Initial population of foxes (2)}
\]

\[
y_2 = \text{Population of rabbits}
\]

\[
y_{i2} = \text{Initial population of rabbits (5)}
\]

1. Use \( \text{MODE} \) to set \( \text{Graph} = \text{DIFF EQUATIONS} \).

2. In the Y= Editor (\( \downarrow [Y=] \)), define the differential equations and enter the initial conditions.

   \text{Note:} \text{ To speed up graphing times, clear any other equations in the Y= Editor. With FLDOFF, all equations are evaluated even if they are not selected.}

3. Press:
   \( \text{F1} \quad 9 \)
   — or —
   \( \text{F1} \quad \text{Set Axes = ON, Labels = ON, Solution Method = RK, and Fields = FLDOFF.} \)
4. In the Y= Editor, press:

2nd [F7] Set Axes = TIME.

5. In the Window Editor ([WINDOW]), set the Window variables.

\[
\begin{align*}
t0 &= 0 & \quad x_{\text{min}} &= -1 & \quad n_{\text{curves}} &= 0 \\
t_{\text{max}} &= 10 & \quad x_{\text{max}} &= 10 & \quad \text{diftol} &= .001 \\
t_{\text{step}} &= \pi/24 & \quad x_{\text{scl}} &= 5 \\
t_{\text{plot}} &= 0 & \quad y_{\text{min}} &= -10 \\
 & & \quad y_{\text{max}} &= 40 \\
 & & \quad y_{\text{scl}} &= 5
\end{align*}
\]

6. Graph the differential equations ([GRAPH]).

7. Press F3 to trace. Then press 3 ENTER to see the number of foxes (yc for y1) and rabbits (yc for y2) at \( t=3 \).

**Note:** Use $\ominus$ and $\oslash$ to move the trace cursor between the curves for y1 and y2.
8. Return to the Y= Editor. Press:

F1 9
— or —

Set Fields = DIRFLD.

Note: In this example, DIRFLD is used for two related differential equations that do not represent a 2nd-order equation.

9. Press:

2nd [F7]

Confirm that the axes are set as shown.

10. In the Y= Editor, clear the initial conditions for \( y_{i1} \) and \( y_{i2} \).

11. Return to the Graph screen, which displays only the direction field.

12. To graph a family of solutions, return to the Y= Editor and enter the initial conditions shown below.

\[ y_{i1} = \{2, 6, 7\} \] and \[ y_{i2} = \{5, 12, 18\} \]

Note: Use a list to specify more than one initial condition.
Consider a logistic growth model \( \frac{dP}{dt} = 0.001P(100-P) \), with the initial condition \( P(0) = 10 \). Use the `BldData` instruction to compare the graphing points calculated by the RK and Euler solution methods. Then plot those points along with a graph of the equation’s exact solution.

**Example Comparison of RK and Euler**

Because \( t_0=0 \) and \( t_{\text{max}}=10 \), you can trace in the range \( 0 \leq t \leq 10 \).

**Note:** Use \( \uparrow \) and \( \downarrow \) to move the trace cursor from one initial condition curve to another.

**Example**

1. Press `[MODE]` and set `Graph=DIFF EQUATIONS`. 
2. Express the 1st-order equation in terms of \( y_1' \) and \( y_1 \).

\[
y_1' = 0.001 y_1(100 - y_1)
\]

Do not use implied multiplication between the variable and parentheses. If you do, it is treated as a function call.

3. Enter the equation in the Y= Editor (\( \text{[Y=]} \)).

4. Enter the initial condition:

\( y_{i1} = 10 \)

5. Press:

\( \text{F1} \ 9 \)

—or—

\( \text{F} \)

Set Solution Method = RK and Fields = FLDOFF.

**Note:** To speed up graphing times, clear any other equations in the Y= Editor. With FLDOFF, all equations are evaluated even if they are not selected.

6. In the Window Editor (\( \text{[WINDOW]} \)), set the Window variables.
Important: Change tstep from .1 (its default) to 1. Otherwise, BldData calculates too many rows for the data variable and a Dimension error occurs.

7. In the Home screen

use BldData to create a data variable containing the RK graphing points.

BldData rklog

8. Return to the Y= Editor, press:

Set Solution Method = EULER.

Note: You do not need to graph the equation before using BldData. For more information about BldData, refer to the Technical Reference module.

9. Return to the Home screen, and use BldData to create a data variable containing the Euler graphing points.

BldData eulerlog
10. Use the Data/Matrix Editor (APPS) to create a new data variable named errorlog.

**Note:** errorlog lets you combine the data in rklog and eulerlog so that you can view the two sets of data side by side.

11. In this new data variable, define the c1, c2, and c3 column headers to refer to data in rklog and eulerlog. Also, enter column titles as shown.

To define a column header, move the cursor to that column, press F4, type the reference expression (such as rklog[1] for c1), and press ENTER.

**Note:** rklog[1] and rklog[2] refer to column 1 and 2 in rklog, respectively. Likewise with eulerlog[2].

12. In the Data/Matrix Editor, press F2. Then press F1 and define Plot 1 for the RK data, as shown to the right.

13. Define Plot 2 for the Euler data. Use the values shown to the right.

14. Return to the Y= Editor, press MODE, and set Graph = FUNCTION.
15. The exact solution to the differential equation is given below. Enter it as y1.

\[ y_1 = \frac{100 \cdot e^{x/10}}{e^{x/10} + 9} \]

**Note:** You can use `deSolve()` to find this exact, general solution.

16. In the Window Editor, set the Window variables.

- `xmin = -10`
- `ymin = -10`
- `xres = 2`
- `xmax = 100`
- `ymax = 120`
- `xscl = 10`
- `yscl = 10`

17. Display the Graph screen (`[GRAPH]`).

**Note:** The fuzzy line on the graph indicates differences between the **RK** and **Euler** values.

18. In the Window Editor, set the Window variables to zoom in so that you can examine the differences in more detail.

- `xmin = 39.7`
- `ymin = 85.5`
- `xres = 2`
- `xmax = 40.3`
- `ymax = 86`
- `xscl = 0.1`
- `yscl = 0.1`

19. Return to the Graph screen.
20. Press \( F_3 \) to trace, and then press \( \triangleright \) or \( \triangleleft \) until \( y_1 \) is selected. (1 shows in upper right corner.) Then enter 40.

By moving the trace cursor to trace each solution to \( xc = 40 \), you can find that:

- The exact solution (\( y_1 \)) is 85.8486, rounded to six digits.
- The RK solution (Plot 1) is 85.8952.
- The Euler solution (Plot 2) is 85.6527.

You can also use the Data/Matrix Editor to open the errorlog data variable and scroll to time = 40.

**Example of the deSolve( ) Function**

The deSolve( ) function lets you solve many 1st- and 2nd-order ordinary differential equations exactly.
**Example**

For a general solution, use the following syntax. For a particular solution, refer to the Technical Reference module.

\[ \text{deSolve}(1stOr2ndOrderODE, \text{independentVar}, \text{dependentVar}) \]

Using the logistic 1st-order differential equation, find the general solution for \( y \) with respect to \( t \).

\[ \text{deSolve}(y' = 1/1000 \cdot y \cdot (100-y), t, y) \]

For ', type \([2nd][']\).

Do not use implied multiplication between the variable and parentheses. If you do, it will be treated as a function call.

**Notes:**

- For maximum accuracy, use \( 1/1000 \) instead of \( .001 \). A floating-point number can introduce round-off errors.
- This example does not involve graphing, so you can use any Graph mode.
Before using `deSolve( )`, clear any existing `t` and `y` variables. Otherwise, an error occurs.

1. In the Home screen
   - Use `deSolve( )` to find the general solution.

2. Use the solution to define a function.
   a) Press to highlight the solution in the history area. Then press to autopaste it into the entry line.
   b) Insert the `Define` instruction at the beginning of the line. Then press .

Note: Press `2nd` to move to the beginning of the entry line.

3. For an initial condition `y=10` with `t=0`, use `solve( )` to find the `@1` constant.

Note: If you got a different constant (`@2`, etc.), solve for that constant.
4. Evaluate the general solution \( y \) with the constant \( @1 = \frac{9}{100} \) to obtain the particular solution shown.

\[
\begin{align*}
y(t) &= \frac{9}{100} t + \left(100 e^{\frac{t}{10}} - 10 \right) e^{\frac{t}{10}} + 9
\end{align*}
\]

You can also use `deSolve()` to solve this problem directly. Enter:

```
deSolve(y' = 1/1000 y*(100-y) and y(0)=10,t,y)
```

## Troubleshooting with the Fields Graph Format

If you have difficulties graphing a differential equation, this section can help you correct the problem. Many problems may be related to your Fields graph format setting.

### Setting the Fields Graph Format

From the Y= Editor, Window Editor, or Graph screen, press:

- \[ F1 \ 9 \]
  
  — or —

- \[ y= \ 1 \]

### What Order Equation Are You Graphing?

<table>
<thead>
<tr>
<th>If the equation is:</th>
<th>Valid Fields settings are:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st-order</td>
<td>SLPFLD or FLDOFF</td>
</tr>
</tbody>
</table>
If the equation is: | Valid Fields settings are:
---|---
2nd-order (system of two 1st-order equations) | DIRFLD or FLDOFF
3rd- or higher-order (system of three or more 1st-order equations) | FLDOFF

Because **Fields = SLPFLD** is the default setting, a common error message is shown to the right.

When you see this or any other error message:

- For your order of equation, use the previous table to find the valid Fields settings. Change to the applicable setting.
- For a particular Fields setting, check the following for information that applies to that setting.
In the Y= Editor

Use F4 to select one and only one 1st-order equation. You can enter multiple equations, but only one at a time can be selected. The selected equation must not refer to any other equation in the Y= Editor. For example:

If \( y_1' = y_2 \), an Undefined variable error occurs when you graph.

In the Graph screen

If the slope field is drawn but no solution curve is plotted, specify an initial condition.
In the **Y= Editor** Enter a valid system of two 1st-order equations. For information about defining a valid system for a 2nd-order equation, refer to Example of a 2nd-Order Equation. 

Set **Axes = CUSTOM:**

If **Axes = TIME**, an Invalid axes error occurs when you graph.

If you enter initial conditions in the Y= Editor, the equations referenced by the custom axes must have the same number of initial conditions. Otherwise, a Dimension error occurs when you graph.

With custom axes Set axes that are valid for your system of equations.

Do not select t for either axis. Otherwise, an Invalid axes error occurs when you graph. 

The two axes must refer to different equations in your system of equations. For example, y1 vs. y2 is valid, but y1 vs. y1' gives an Invalid axes error.
In the Graph screen

If the direction field is drawn but no curve is plotted, enter initial conditions in the Y= Editor or select one interactively from the Graph screen. If you did enter initial conditions, select **ZoomFit:**

![ZoomFit](image)

The **ncurves** Window variable is ignored with DIRFLD. Default curves are not drawn automatically.

**Notes**

With DIRFLD, the equations referenced by the custom axes determine which equations are graphed, regardless of which equations are selected in the Y= Editor. If your system of equations refers to t, the direction field (not the plotted curves) is drawn with respect to one particular time, which is set by the dtime Window variable.

---

**Fields=FLDOFF**

**In the Y= Editor**

If you enter a 2nd- or higher-order equation, enter it as a valid system of equations. All equations (selected or not) must have the same number of initial conditions. Otherwise, a Dimension error occurs when you graph.

![Dimension Error](image)

To set **Axes = TIME** or **CUSTOM**, press:

![Axes Setting](image)
If You Use the Table Screen to View Differential Equations

You can use the Table screen to view the points for a differential equation graph. However, the table may show different equations than those graphed. The table shows only the selected equations, regardless of whether those equations will be plotted with your current **Fields** and **Axes** settings.
Tables

Overview of Steps in Generating a Table

To generate a table of values for one or more functions, use the general steps shown below. For specific information about setting table parameters and displaying the table, refer to the following pages.

Generating a Table

1. Set **Graph** mode and, if necessary, **Angle** mode (MODE).
   
   **Note:** Tables are not available in **3D Graph** mode.

2. Define functions on Y= Editor ([Y=]).

3. Select ([F4]) which defined functions to display in the table.
   
   **Note:** For information on defining and selecting functions with the Y= Editor, refer to *Basic Function Graphing*.

4. Set up the initial table parameters ([TBLSET]).
   
   **Note:** You can specify an automatic table that is based on initial values or that matches a graph, or a manual (ask) table.
5. Display the table ([TABLE]).

**Exploring the Table**

From the Table screen, you can:

- Scroll through the table to see values on other pages.
- Highlight a cell to see its full value.
- Change the table’s setup parameters. By changing the starting or incremental value used for the independent variable, you can zoom in or out on the table to see different levels of detail.
- Change the cell width.
- Edit selected functions.
- Build or edit a manual table to show only specified values of the independent variable.

**Setting Up the Table Parameters**

To set up the initial parameters for a table, use the TABLE SETUP dialog box. After the table is displayed, you can also use this dialog box to change the parameters.
### Displaying the TABLE SETUP Dialog Box

To display the TABLE SETUP dialog box, press [TBLSET]. From the Table screen, you can also press F2.

![TABLE SETUP](image)

<table>
<thead>
<tr>
<th>Setup Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tblStart</td>
<td>If Independent = AUTO and Graph &lt;-&gt; Table = OFF, this specifies the starting value for the independent variable.</td>
</tr>
<tr>
<td>Δtbl</td>
<td>If Independent = AUTO and Graph &lt;-&gt; Table = OFF, this specifies the incremental value for the independent variable. Δtbl can be positive or negative, but not zero.</td>
</tr>
</tbody>
</table>
| Graph <-> Table | If Independent = AUTO:  
OFF — The table is based on the values you enter for tblStart and Δtbl.  
ON — The table is based on the same independent variable values that are used to graph the functions on the Graph screen. These values depend on the Window variables set in the Window Editor and the split screen size. |
### Which Setup Parameters to Use

<table>
<thead>
<tr>
<th>To generate:</th>
<th>tblStart</th>
<th>Δtbl</th>
<th>Graph &lt; - &gt; Table</th>
<th>Independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>An automatic table</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Based on initial values</td>
<td>value</td>
<td>value</td>
<td>OFF</td>
<td>AUTO</td>
</tr>
<tr>
<td>• That matches Graph screen</td>
<td>–</td>
<td>–</td>
<td>ON</td>
<td>AUTO</td>
</tr>
<tr>
<td>A manual table</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>ASK</td>
</tr>
</tbody>
</table>

**Note:** “–” means that any value entered for this parameter is ignored for the indicated type of table.

In SEQUENCE graphing mode, use integers for `tblStart` and `Δtbl`.

### Changing the Setup Parameters

From the **TABLE SETUP** dialog box:
1. Use and to highlight the value or setting to change.
2. Specify the new value or setting.

<table>
<thead>
<tr>
<th>To change:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>tblStart</td>
<td>Type the new value. The existing value is erased when you start to type.</td>
</tr>
<tr>
<td>or</td>
<td>— or —</td>
</tr>
<tr>
<td>Δtbl</td>
<td>Press 4 or 5 to remove the highlighting. Then edit the existing value.</td>
</tr>
<tr>
<td>Graph &lt; − &gt; Table or Independent</td>
<td>Press 4 or 5 to display a menu of valid settings. Then either:</td>
</tr>
<tr>
<td></td>
<td>• Move the cursor to highlight the setting and press [ENTER].</td>
</tr>
<tr>
<td></td>
<td>— or —</td>
</tr>
<tr>
<td></td>
<td>• Press the number for that setting.</td>
</tr>
</tbody>
</table>

**Note:** To cancel a menu or exit the dialog box without saving any changes, press [ESC] instead of [ENTER].

3. After changing all applicable values or settings, press [ENTER] to save your changes and close the dialog box.

**From the Home Screen or a Program**

You can set up a table’s parameters from the Home screen or a program. You can:

- Store values directly to the system variables **tblStart** and **Δtbl**. Refer to “Storing and Recalling Variable Values” in *Operating the Calculator*.
• Set Graph < - > Table and Independent by using the setTable function. Refer to the Technical Reference module.

### Displaying an Automatic Table

If Independent = AUTO on the TABLE SETUP dialog box, a table is generated automatically when you display the Table screen. If Graph < - > Table = ON, the table matches the trace values from the Graph screen. If Graph < - > Table = OFF, the table is based on the values you entered for tblStart and Δtbl.

### Before You Begin

Define and select the applicable functions on the Y= Editor (Y=). This example uses $y_1(x) = x^3 - x/3$.

Then enter the initial table parameters (TblSet).

![TABLE SETUP Table]

### Displaying the Table Screen

To display the Table screen, press [TABLE] or [APPs] 5.

The cursor initially highlights the cell that contains the starting value of the independent variable. You can move the cursor to any cell that contains a value.
First column shows values of the independent variable.

Other columns show corresponding values of the functions selected in the Y= Editor.

Header row shows names of independent variable (x) and selected functions (y1).

Entry line shows full value of highlighted cell.

Note: You can scroll back from the starting value by pressing \( \leftarrow \) or \( \text{2nd} \ \leftarrow \).

To move the cursor: Press:

<table>
<thead>
<tr>
<th>One cell at a time</th>
<th>( \downarrow, \uparrow, \leftarrow, \rightarrow )</th>
</tr>
</thead>
<tbody>
<tr>
<td>One page at a time</td>
<td>( \text{2nd} ) and then ( \downarrow, \uparrow, \leftarrow, \rightarrow )</td>
</tr>
</tbody>
</table>

The header row and the first column are fixed so that they cannot scroll off the screen.

- When you scroll down or up, the variable and function names are always visible across the top of the screen.
- When you scroll right or left, the values of the independent variable are always visible along the left side of the screen.
Changing the Cell Width

Cell width determines the maximum number of digits and symbols (decimal point, minus sign, and “e” for scientific notation) that can be displayed in a cell. All cells in the table have the same width.

Note: By default, the cell width is 6.

To change the cell width from the Table screen:

1. Press **F1 9**
   — or —
   ![Formats dialog box]
   
2. Press **or** to display a menu of valid widths (3–12).

3. Move the cursor to highlight a number and press **[ENTER]**. (For single-digit numbers, you can type the number and press **[ENTER]**.)

4. Press **[ENTER]** to close the dialog box and update the table.

How Numbers Are Displayed in a Cell

Whenever possible, a number is shown according to the currently selected display modes (Display Digits, Exponential Format, etc.). The number may be rounded as necessary. However:
• If a number’s magnitude is too large for the current cell width, the number is rounded and shown in scientific notation.

• If the cell width is too narrow even for scientific notation, “...” is shown.

Notes:

• If a function is undefined at a particular value, undef is displayed in the cell.

• Use MODE to set the display modes.

By default, Display Digits = FLOAT 6. With this mode setting, a number is shown with up to six digits, even if the cell is wide enough to show more. Other settings similarly affect a displayed number.

<table>
<thead>
<tr>
<th>If cell width is:</th>
<th>Full Precision</th>
<th>3</th>
<th>6</th>
<th>9</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Precision</td>
<td>1.2345678901</td>
<td>1.2</td>
<td>1.2346</td>
<td>1.23457</td>
<td>1.23457*</td>
</tr>
<tr>
<td>-123456.78</td>
<td>...</td>
<td>-1.2E5</td>
<td>-123457.</td>
<td>-123457.*</td>
<td></td>
</tr>
<tr>
<td>.000005</td>
<td>...</td>
<td>5.E-6</td>
<td>.000005</td>
<td>.000005</td>
<td></td>
</tr>
<tr>
<td>1.2345678E19</td>
<td>...</td>
<td>1.2E19</td>
<td>1.2346E19</td>
<td>1.23457E19*</td>
<td></td>
</tr>
<tr>
<td>-1.23456789012E-200</td>
<td>...</td>
<td>...</td>
<td>-1.2E-200</td>
<td>-1.2346E-200*</td>
<td></td>
</tr>
</tbody>
</table>

*Note: Depending on display mode settings, some values are not shown in full precision even when the cell is wide

Note: To see a number in full precision, highlight the cell and look at the entry line.
If Results are Complex Numbers

A cell shows as much as possible of a complex number (according to the current display modes) and then shows “...” at the end of the displayed portion.

When you highlight a cell containing a complex number, the entry line shows the real and imaginary parts with a maximum of four digits each (FLOAT 4).

Editing a Selected Function

From a table, you can change a selected function without having to use the Y= Editor.

1. Move the cursor to any cell in the column for that function. The table’s header row shows the function names (y1, etc.).
2. Press \( \text{F4} \) to move the cursor to the entry line, where the function is displayed and highlighted.
   
   **Note:** You can use this feature to view a function without leaving the table.
3. Make any changes, as necessary.
   - Type the new function. The old function is erased when you begin typing.
     — or —
   - Press \( \text{CLEAR} \) to clear the old function. Then type the new one.
     — or —
   - Press \( \text{ã} \) or \( \text{ø} \) to remove the highlighting. Then edit the function.
   
   **Note:** To cancel any changes and return the cursor to the table, press \( \text{ESC} \) instead of \( \text{ENTER} \).
4. Press \textbf{ENTER} to save the edited function and update the table. The edited function is also saved in the Y= Editor.

\textbf{If You Want to Change the Setup Parameters}

After generating an automatic table, you can change its setup parameters as necessary.

Press \textbf{F2} or \textbf{[TBLSET]} to display the TABLE SETUP dialog box. Then make your changes.

\textbf{Building a Manual (Ask) Table}

If \texttt{Independent = ASK} on the TABLE SETUP dialog box, the TI-89 Titanium lets you build a table manually by entering specific values for the independent variable.

\textbf{Displaying the Table Screen}

To display the Table screen, press \textbf{[TABLE]}.

If you set \texttt{Independent = ASK} (with \textbf{[TBLSET]}) before displaying a table for the first time, a blank table is displayed. The cursor highlights the first cell in the independent variable column.
If you first display an automatic table and then change it to **Independent = ASK**, the table continues to show the same values. However, you can no longer see additional values by scrolling up or down off the screen.

**Entering or Editing an Independent Variable Value**

You can enter a value in column 1 (independent variable) only.

1. Move the cursor to highlight the cell you want to enter or edit.
   - If you start with a blank table, you can enter a value in consecutive cells only (row 1, row 2, etc.). You cannot skip cells (row 1, row 3).
   - If a cell in column 1 contains a value, you can edit that value.
2. Press **F3** to move the cursor to the entry line.
3. Type a new value or expression, or edit the existing value.
4. Press **ENTER** to move the value to the table and update the corresponding function values.

   **Note:** To enter a new value in a cell, you do not need to press **F3**. Simply begin typing.

The cursor returns to the entered cell. You can use ☐ to move to the next row.
Note: In this example, you can move the cursor to column 2, but you can enter values in column 1 only.

**Entering a List in the Independent Variable Column**

1. Move the cursor to highlight any cell in the independent variable column.
2. Press `F4` to move the cursor to the entry line.
3. Type a series of values, enclosed in braces `{ }` and separated by commas. For example:

   \[ x = \{1, 1.5, 1.75, 2\} \]

   You can also enter a list variable or an expression that evaluates to a list.

   **Note:** If the independent variable column contains existing values, they are shown as a list (which you can edit).

4. Press `ENTER` to move the values into the independent variable column. The table is updated to show the corresponding function values.
Adding, Deleting, or Clearing

<table>
<thead>
<tr>
<th>To:</th>
<th>Do this:</th>
</tr>
</thead>
</table>
| Insert a new row above a specified row | Highlight a cell in the specified row and press:  
  @  2nd [F6]  
  The new row is undefined (undef) until you enter a value for the independent variable. |
| Delete a row | Highlight a cell in the row and press [F5]. If you highlight a cell in the independent variable column, you can also press ←. |
| Clear the entire table (but not the selected Y= functions) | Press [F1] 8. When prompted for confirmation, press [ENTER]. |

Cell Width and Display Formats

Several factors affect how numbers are displayed in a table.

From the Home Screen or a Program

System variable tblInput contains a list of all independent variable values entered in the table, even those not currently displayed. tblInput is also used for an automatic table, but it contains only the independent variable values that are currently displayed.

Before displaying a table, you can store a list of values directly to the tblInput system variable.
Additional Graphing Topics

Collecting Data Points from a Graph

From the Graph screen, you can store sets of coordinate values and/or math results for later analysis. You can store the information as a single-row matrix (vector) on the Home screen or as data points in a system data variable that can be opened in the Data/Matrix Editor.

Collecting the Points

1. Display the graph. (This example shows \( y_1(x) = 5\cos(x) \).)
2. Display the coordinates or math results you want to collect.
3. Save the information to the Home screen or the \( sysData \) variable. \( \downarrow \) \( \langle \rangle \) (Home screen) or \( \downarrow \) \( \langle \rangle \) (\( sysData \) variable)
4. Repeat the process as necessary.

Note: To display coordinates or math results, trace a function with \( \text{F3} \) or perform an \( \text{F5} \) Math operation (such as Minimum or Maximum). You can also use the free-moving cursor.
Displayed coordinates are added to the Home screen’s history area (but not the entry line) as a single-row matrix or vector.

Displayed coordinates are stored in a data variable named sysData, which you can open in the Data/Matrix Editor.

Note: Use a split screen to show a graph and the Home screen or Data/Matrix Editor at the same time.

Notes about SysData Variable

- When you press: 
  - If sysData does not exist, it is created in the MAIN folder.
  - If sysData already exists, new data is appended to the end of any existing data.
    Existing titles or column headers (for the affected columns) are cleared; titles are replaced with the applicable titles for the new data.

- The sysData variable can be cleared, deleted, etc., just as any other data variable. However, it cannot be locked.
• If the Graph screen contains a function or stat plot that references the current contents of sysData, this command will not operate.

**Graphing a Function Defined on the Home Screen**

In many cases, you may create a function or expression on the Home screen and then decide to graph it. You can copy an expression to the Y= Editor, or graph it directly from the Home screen without using the Y= Editor.

**What Is the “Native” Independent Variable?**

On the Y= Editor, all functions must be defined in terms of the current graph mode’s “native” independent variable.

<table>
<thead>
<tr>
<th>Graph Mode</th>
<th>Native Independent Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>x</td>
</tr>
<tr>
<td>Parametric</td>
<td>t</td>
</tr>
<tr>
<td>Polar</td>
<td>θ</td>
</tr>
<tr>
<td>Sequence</td>
<td>n</td>
</tr>
<tr>
<td>3D</td>
<td>x, y</td>
</tr>
<tr>
<td>Differential Equation</td>
<td>t</td>
</tr>
</tbody>
</table>
## Copying from the Home Screen to the Y= Editor

If you have an expression on the Home screen, you can use any of the following methods to copy it to the Y= Editor.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| Copy and paste       | 1. Highlight the expression on the Home screen. Press [F1] and select 5:Copy.  
2. Display the Y= Editor, highlight the desired function, and press [ENTER].  
**Note:** Instead of using [F1] 5 or [F1] 6 to copy and paste, use: [COPY] or [PASTE] |
| STO►                | Store the expression to a Y= function name.  
\[
2x^3+3x^2-4x+12 \rightarrow y_1(x) 
\]  
Use the complete function name: \( y_1(x) \), not just \( y_1 \).  
**Note:** To copy an expression from the Home screen’s history area to the entry line, use the auto-paste feature or copy and paste. |
| Define command       | Define the expression as a user-defined Y= function.  
\[
\text{Define } y_1(x)=2x^3+3x^2-4x+12 
\]  
**Note:** Define is available from the Home screen’s [F4] toolbar menu. |
### Method | Description
--- | ---
2nd [RCL] | If the expression is already stored to a variable:
1. Display the Y= Editor, highlight the desired function, and press [ENTER].
2. Press 2nd [RCL]. Type the variable name that contains the expression, and press [ENTER] twice.
**Important:** To recall a function variable such as $f_1(x)$, type only $f_1$, not the full function name.
3. Press [ENTER] to save the recalled expression in the Y= Editor’s function list.
**Note:** 2nd [RCL] is useful if an expression is stored to a variable or function that does not correspond to the Y= Editor, such as $a_1$ or $f_1(x)$.

---

**Graphing Directly from the Home Screen**

The **Graph** command lets you graph an expression from the Home screen without using the Y= Editor. Unlike the Y= Editor, **Graph** lets you specify an expression in terms of any independent variable, regardless of the current graphing mode.

<table>
<thead>
<tr>
<th>If the expression is in terms of:</th>
<th>Use the Graph command as shown in this example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The native independent variable</td>
<td>Graph $1.25x \times \cos(x)$</td>
</tr>
<tr>
<td></td>
<td>For function graphing, $x$ is the native variable.</td>
</tr>
</tbody>
</table>
### Additional Graphing Topics

#### If the expression is in terms of:

<table>
<thead>
<tr>
<th>Use the Graph command as shown in this example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A non-native independent variable</td>
</tr>
</tbody>
</table>

Specify the independent variable; otherwise, you may get an error.

---

**Note:** Graph uses the current Window variable settings and is available from the Home screen’s F4 toolbar menu.

Graph does not work with sequence graphs or differential equations. For parametric, polar, and 3D graphs, use the following variations.

In PARAMETRIC graphing mode: \[ \text{Graph } xExpr, yExpr, t \]

In POLAR graphing mode: \[ \text{Graph } expr, \theta \]

In 3D graphing mode: \[ \text{Graph } expr, x, y \]

**Note:** To create a table from the Home screen, use the Table command. It is similar to Graph. Both share the same expressions.

Graph does not copy the expression to the Y= Editor. Instead, it temporarily suspends any functions selected on the Y= Editor. You can trace, zoom, or show and edit Graph expressions on the Table screen, just the same as Y= Editor functions.

---

**Clearing the Graph Screen**

Each time you execute Graph, the new expression is added to the existing ones. To clear the graphs:
• Execute the **ClrGraph** command (available from the Home screen’s **F4 Other** toolbar menu).

  – or –

• Display the Y= Editor. The next time you display the Graph screen, it will use the functions selected on the Y= Editor.

### Extra Benefits of User-Defined Functions

You can define a user-defined function in terms of any independent variable. For example:

```
Define f1(aa)=1.25aa cos(aa).
```

Define **f1(aa)=1.25aa*cos(aa)**

Graph **f1(x)**

Refers to the function by using the native independent variable.

and:

```
Define f1(aa)=1.25aa*cos(aa)
f1(x)→y1(x)
```

### Graphing a Piecewise Defined Function

To graph a piecewise function, you must first define the function by specifying boundaries and expressions for each piece. The **when** function is extremely useful for
two-piece functions. For three or more pieces, it may be easier to create a multi-
statement, user-defined function.

**Using the When Function**

To define a two-piece function, use the syntax:

\[
\text{when}(\text{condition}, \text{trueExpression}, \text{falseExpression})
\]

For example, suppose you want to graph a function with two pieces.

<table>
<thead>
<tr>
<th>When:</th>
<th>Use expression:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x &lt; 0)</td>
<td>(-x)</td>
</tr>
<tr>
<td>(x \geq 0)</td>
<td>(5 \cos(x))</td>
</tr>
</tbody>
</table>

In the Y= Editor: The function is “pretty printed” in this form. Enter the function in this form.

\[
y1 = \begin{cases} 
-x, & x < 0 \\
5 \cos(x), & \text{else} 
\end{cases}
\]

For three or more pieces, you can use nested \textit{when} functions.
Note: To enter \texttt{when}, type it or use the \texttt{CATALOG}.

<table>
<thead>
<tr>
<th>When:</th>
<th>Use expression:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x &lt; -\pi )</td>
<td>( 4 \sin(x) )</td>
</tr>
<tr>
<td>( x \geq -\pi ) and ( x &lt; 0 )</td>
<td>( 2x + 6 )</td>
</tr>
<tr>
<td>( x \geq 0 )</td>
<td>( 6 - x^2 )</td>
</tr>
</tbody>
</table>

In the \texttt{Y= Editor}:

\[
y_1(x) = \begin{cases} 
4 \sin(x), & x < -\pi, x < 0 \\
2x + 6, & \text{else} \\
6 - x^2, & \text{else}
\end{cases}
\]

where:

\[
y_1(x) = \text{when}(x<0, \text{when}(x<\pi, 4 \sin(x), 2x+6), 6-x^2)
\]

This nested function is in effect when \( x<0 \).

Nested functions quickly become complex and difficult to visualize.

\textbf{Using a Multi-Statement, User-Defined Function}

For three or more pieces, you may want to create a multi-statement, user-defined function.
For example, consider the previous three-piece function.

<table>
<thead>
<tr>
<th>When:</th>
<th>Use expression:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x &lt; -\pi$</td>
<td>$4 \sin(x)$</td>
</tr>
<tr>
<td>$x \geq -\pi$ and $x &lt; 0$</td>
<td>$2x + 6$</td>
</tr>
<tr>
<td>$x \geq 0$</td>
<td>$6 - x^2$</td>
</tr>
</tbody>
</table>

**Note:** For information about similarities and differences between functions and programs, refer to *Programming*.

A multi-statement, user-defined function can have many of the control and decision-making structures (**If**, **Elseif**, **Return**, etc.) used in programming. When creating the structure of a function, it may be helpful to visualize it first in a block form.

```plaintext
Func
  If $x < -\pi$ Then
    Return $4 \sin(x)$
  ElseIf $x \geq -\pi$ and $x < 0$ Then
    Return $2x + 6$
  Else
    Return $6 - x^2$
  EndIf
EndFunc
```

**Func** and **EndFunc** must begin and end the function.

When entering a multi-statement function on the Y= Editor or Home screen, you must enter the entire function on a single line.
Use a colon (:) to separate each statement.

```
Func:If x<−π Then:Return 4*sin(x): ... :EndIf:EndFunc
```

In the Y= Editor:

Only **Func** is shown for a multi-statement function.

Enter a multi-statement function on one line. Be sure to include colons.

---

**From the Home Screen or a Program**

From the Home screen, you can also use the **Define** command to create a multi-statement, user-defined function.

Information is available on copying a function from the Home screen to the Y= Editor.

From the Program Editor, you can create a user-defined function. For example, use the Program Editor to create a function named `f1(xx)`. In the Y= Editor, set `y1(x) = f1(x)`.

**Graphing a Family of Curves**

By entering a list in an expression, you can plot a separate function for each value in the list. (You cannot graph a family of curves in SEQUENCE or 3D graphing mode.)
Examples Using the Y= Editor

Enter the expression \(\{2,4,6\} \sin(x)\) and graph the functions.

Note: Enclose list elements in braces (\(2^{\text{nd}}[\{]\) and \(2^{\text{nd}}[\}]\) and separate them with commas.

Enter the expression \(\{2,4,6\} \sin(\{1,2,3\} \times x)\) and graph the functions.

Note: The commas are shown in the entry line but not in the function list.

Example Using the Graph Command

Similarly, you can use the Graph command from the Home screen or a program.
Simultaneous Graphs with Lists

When the graph format is set for **Graph Order = SIMUL**, the functions are graphed in groups according to the element number in the list.

<table>
<thead>
<tr>
<th>y₁=₂sin(x)</th>
<th>y₂=₄sin(x)</th>
<th>y₅=cos(x)</th>
</tr>
</thead>
</table>

For these example functions, the TI-89 Titanium / Voyage™ 200 Graphing Calculator graphs three groups.

- 2 sin(x), x+4, cos(x)
- 4 sin(x), 2x+4
- 6 sin(x), 3x+4

The functions within each group are graphed simultaneously, but the groups are graphed sequentially.

**Note:** To set graph formats from the Y= Editor, Window Editor, or Graph screen, press: `×` `1`

When Tracing a Family of Curves

Pressing ⊞ or ⊜ moves the trace cursor to the next or previous curve in the same family before moving to the next or previous selected function.
Using the Two-Graph Mode

In two-graph mode, the calculator’s graph-related features are duplicated, giving you two independent graphing calculators. The two-graph mode is only available in split screen mode. For more information about split screens, refer to Split Screens.

Setting the Mode

Several mode settings affect the two-graph mode, but only two settings are required. Both are on Page 2 of the MODE dialog box.

1. Press MODE. Then press F2 to display Page 2.

2. Set the following required modes.
   - Split Screen = TOP-BOTTOM or LEFT-RIGHT
   - Number of Graphs = 2

3. Optionally, you can set the following modes.

   Page 1:  • Graph = Graph mode for top or left side of the split
4. Press [ENTER] to close the dialog box.

**The Two-Graph Screen**

A two-graph screen is similar to a regular split screen.

**Independent Graph-Related Features**

Both Graph 1 and Graph 2 have independent:

- **Graph modes (FUNCTION, POLAR, etc.).** Other modes such as **Angle, Display Digits**, etc., are shared and affect both graphs.
• Window Editor variables.
• Table setup parameters and Table screens.
• Graph formats such as **Coordinates**, **Axes**, etc.
• Graph screens.
• **Y=** Editors. However, both graphs share common function and stat plot definitions. **Note:** The **Y=** Editor is completely independent only when the two sides use different graphing modes (as described below).

Independent graph-related applications (**Y=** Editor, Graph screen, etc.) can be displayed on both sides of the screen at the same time.

Non-graph-related applications (Home screen, Data/Matrix Editor, etc.) are shared and can be displayed on only one side at a time.

**The **Y=** Editor in Two-Graph Mode**

Even in two-graph mode, there is actually only one **Y=** Editor, which maintains a single function list for each Graph mode setting. However, if both sides use the same graphing mode, each side can select different functions from that single list.

• When both sides use different graphing modes, each side shows a different function list.
• When both sides use the same graphing mode, each side shows the same function list.
  - You can use [F4] to select different functions and stat plots (indicated by ✔) for each side.
  - If you set a display style for a function, that style is used by both sides.[2nd] [F6]

• Suppose Graph 1 and Graph 2 are set for function graphing. Although both sides show the same function list, you can select (✔) different functions for graphing.

**Note:** If you make a change on the active Y= Editor (redefine a function, change a style, etc.), that change is not reflected on the inactive side until you switch to it.

**Using a Split Screen**

For more complete information about split screens, refer to *Split Screens*.

• To switch from one graph side to the other, press [2nd] [→-] (second function of [APPS]).

• To display different applications:
  - Switch to the applicable graph side and display the application as you normally would.
  - or –
  - Use [MODE] to change Split 1 App and/or Split 2 App.

• To exit two-graph mode:
- Use [MODE] to set **Number of Graphs = 1**, or exit the split screen by setting **Split Screen = FULL**.

  – or –

- Press [2nd] [QUIT] twice. This always exits a split screen and returns to a full-sized Home screen.

**Note:** You can display non-graph-related applications (such as the Home screen) on only one side at a time.

**Remember that the Two Sides Are Independent**

In two-graph mode, the two sides may appear to be related when, in fact, they are not. For example:

For Graph 1, the Y= Editor lists y(x) functions.

For Graph 2, the polar graph uses r(θ) equations that are not shown.

After the two-graph mode is set up, graph-related operations refer to the active graph side. For example:

```
10→xmax
```

affects either Graph 1 or Graph 2, depending on which is active when you execute the command.
To switch the active sides, press \( \text{2nd} \ [\text{p} \text{p}] \) or use the \textbf{switch} function, \textbf{switch(1)} or \textbf{switch(2)}.

**Drawing a Function or Inverse on a Graph**

For comparison purposes, you may want to draw a function over your current graph. Typically, the drawn function is some variation of the graph. You can also draw the inverse of a function. (These operations are not available for 3D graphs.)

**Drawing a Function, Parametric, or Polar Equation**

Execute \textbf{DrawFunc}, \textbf{DrawParm}, or \textbf{DrawPol} from the Home screen or a program. You cannot draw a function or equation interactively from the Graph screen.

\begin{align*}
\textbf{DrawFunc} & \quad \textit{expression} \\
\textbf{DrawParm} & \quad \textit{expression1}, \textit{expression2} [,\textit{tmin}] [,\textit{tmax}] [,\textit{tstep}] \\
\textbf{DrawPol} & \quad \textit{expression} [,\textit{\theta min}] [,\textit{\theta max}] [,\textit{\theta step}] \\
\end{align*}

For example:

1. Define \( y_1(x) = 0.1x^3 - 2x + 6 \) on the Y= Editor, and graph the function.

![Graph of y1(x) = 0.1x^3 - 2x + 6]
2. On the Graph screen, press: \texttt{[2nd} [F6] and select \texttt{2:DrawFunc}.

To display the Home screen and put \texttt{DrawFunc} in the entry line, press: \texttt{[2nd} [F6] 2

3. On the Home screen, specify the function to draw.

4. Press \texttt{[ENTER]} to draw the function on the Graph screen.

You cannot trace, zoom, or perform a math operation on a drawn function.

\textbf{Note}: To clear the drawn function, press \texttt{[F4]}
– or –
\texttt{[2nd} [F6] and select \texttt{1:ClrDraw}

\textbf{Drawing the Inverse of a Function}

Execute \texttt{DrawInv} from the Home screen or a program. You cannot draw an inverse function interactively from the Graph screen.

\texttt{DrawInv \textit{expression}}

For example, use the graph of \( y_1(x) = 0.1x^3 - 2x + 6 \) as shown above.
1. On the Graph screen, press: \[2^{\text{nd}}\quad [F6]\] and select \textbf{3:DrawInv}
   
   To display the Home screen and put \textbf{DrawInv} in the entry line, press:
   \[2^{\text{nd}}\quad [F6] \quad 3\]

2. On the Home screen, specify the inverse function.

3. Press \textbf{ENTER}.
   
   The inverse is plotted as \((y,x)\) instead of \((x,y)\).

\textbf{Drawing a Line, Circle, or Text Label on a Graph}

You can draw one or more objects on the Graph screen, usually for comparisons. For example, draw a horizontal line to show that two parts of a graph have the same \(y\) value. (Some objects are not available for 3D graphs.)

\textbf{Clearing All Drawings}

A drawn object is not part of the graph itself. It is drawn “on top of” the graph and remains on the screen until you clear it.
From the Graph screen:

- `2nd [F6]` and select `1:ClrDraw`.
- or –
- Press `F4` to regraph.

**Note:** You can also enter **ClrDraw** on the Home screen’s entry line.

You can also do anything that causes the Smart Graph feature to redraw the graph (such as change the Window variables or deselect a function on the Y= Editor).
Drawing a Point or a Freehand Line

From the Graph screen:

1. Press \texttt{\textasciitilde}\texttt{2nd} [\texttt{F7}] and select \texttt{1:Pencil}.

2. Move the cursor to the applicable location.

<table>
<thead>
<tr>
<th>To draw a:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point (pixel-sized)</td>
<td>Press \texttt{ENTER}.</td>
</tr>
<tr>
<td>Freehand line</td>
<td>Press and hold \texttt{↑}, and move the cursor to draw the line.</td>
</tr>
</tbody>
</table>

\textbf{Note:} When drawing a freehand line, you can move the cursor diagonally.

After drawing the point or line, you are still in \texttt{Pencil} mode.

- To continue drawing, move the cursor to another point.
- To quit, press \texttt{ESC}.

\textbf{Note:} If you start drawing on a white pixel, the pencil draws a black point or line. If you start on a black pixel, the pencil draws a white point or line (which can act as an eraser).
Erasing Individual Parts of a Drawing Object

From the Graph screen:

1. 2nd [F7] and select 2:Eraser. The cursor is shown as a small box.

2. Move the cursor to the applicable location.

<table>
<thead>
<tr>
<th>To erase:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area under the box</td>
<td>Press [Enter].</td>
</tr>
<tr>
<td>Along a freehand line</td>
<td>Press and hold ↑, and move the cursor to erase the line.</td>
</tr>
</tbody>
</table>

**Note:** These techniques also erase parts of graphed functions.

After erasing, you are still in Eraser mode.

- To continue erasing, move the box cursor to another location.
- To quit, press [ESC].

Drawing a Line Between Two Points

From the Graph screen:

1. 2nd [F7] and select 3:Line.
2. Move the cursor to the 1st point, and press [ENTER].
3. Move to the 2nd point, and press [ENTER]. (As you move, a line extends from the 1st point to the cursor.)

   **Note:** Use [2nd] to move the cursor in larger increments; [2nd] ◄, etc.

After drawing the line, you are still in **Line** mode.

- To continue drawing another line, move the cursor to a new 1st point.
- To quit, press [ESC].

### Drawing a Circle

From the Graph screen:

1. [2nd] [F7] and select **4:Circle**.

2. Move the cursor to the center of the circle, and press [ENTER].

3. Move the cursor to set the radius, and press [ENTER].

   **Note:** Use [2nd] to move the cursor in larger increments; [2nd] ◄, etc.
Drawing a Horizontal or Vertical Line

From the Graph screen:

1. 2nd [F7] 
   and select 5:Horizontal or 6:Vertical. A horizontal or vertical line and a flashing cursor are displayed on the screen.

   If the line is initially displayed on an axis, it may be difficult to see. However, you can easily see the flashing cursor.

2. Use the cursor pad to move the line to the appropriate position. Then press ENTER.

   After drawing the line, you are still in “line” mode.

   • To continue, move the cursor to another location.
   • To quit, press ESC.

   Note: Use 2nd to move the cursor in larger increments; 2nd ◄, etc.

Drawing a Tangent Line

To draw a tangent line, use the F5 Math toolbar menu. From the Graph screen:

1. Press F5 and select A:Tangent.
2. As necessary, use \( \Theta \) and \( \Phi \) to select the applicable function.

3. Move the cursor to the tangent point, and press \( \text{ENTER} \).

The tangent line is drawn, and its equation is displayed.

**Note:** To set the tangent point, you can also type its \( x \) value and press \( \text{ENTER} \).

### Drawing a Line Based on a Point and a Slope

To draw a line through a specified point with a specified slope, execute the \texttt{DrawSlp} command from the Home screen or a program. Use the syntax:

\[
\texttt{DrawSlp } x, y, \text{ slope}
\]

You can also access \texttt{DrawSlp} from the Graph screen.

1. \( \textbf{2nd} \ [\text{F6}] \)
   and select 6:\texttt{DrawSlp}. This switches to the Home screen and puts \texttt{DrawSlp} in the entry line.

2. Complete the command, and press \( \text{ENTER} \).

The calculator automatically switches to the \texttt{Graph} screen and draws the line.
Typing Text Labels

From the Graph screen:

1. [2nd] [F7] and select 7:Text.

2. Move the text cursor to the location where you want to begin typing.

3. Type the text label.

   After typing the text, you are still in “text” mode.
   • To continue, move the cursor to another location.
   • To quit, press [ENTER] or [ESC].

   **Note:** The text cursor indicates the upper-left corner of the next character you type.

From the Home Screen or a Program

Commands are available for drawing any of the objects described in this section. There are also commands (such as `PxlOn`, `PxlLine`, etc.) that let you draw objects by specifying exact pixel locations on the screen.

For a list of the available drawing commands, refer to “Drawing on the Graph Screen” in Programming.
Saving and Opening a Picture of a Graph

You can save an image of the current Graph screen in a PICTURE (or PIC) variable. Then, at a later time, you can open that variable and display the image. This saves the image only, not the graph settings used to produce it.

Saving a Picture of the Whole Graph Screen

A picture includes any plotted functions, axes, tick marks, and drawn objects. The picture does not include lower and upper bound indicators, prompts, or cursor coordinates.

Display the Graph screen as you want to save it; and then:

1. Press \( \text{F1} \) and select \( 2:\text{Save Copy As} \).

2. Specify the type (Picture), folder, and a unique variable name.

3. Press \( \text{ENTER} \). After typing in an input box such as Variable, you must press \( \text{ENTER} \) twice.

Important: By default, Type = GDB (for graph database). You must set Type = Picture.
Saving a Portion of the Graph Screen

You can define a rectangular box that encloses only the portion of the Graph screen that you want to save.

1. Press 2nd [F7] and select 8:Save Picture.
   A box is shown around the outer edge of the screen.

   **Note:** You cannot save a portion of a 3D graph.

2. Set the 1st corner of the box by moving its top and left sides. Then press ENTER.
   **Note:** Use ◼ and ◼ to move the top or bottom, and use ◼ and ◼ to move the sides.

3. Set the 2nd corner by moving the bottom and right sides. Then press ENTER.

4. Specify the folder and a unique variable name.

5. Press ENTER. After typing in an input box such as Variable, you must press ENTER twice.

   **Note:** When saving a portion of a graph, Type is automatically fixed as Picture.
Opening a Graph Picture

When you open a graph picture, it is superimposed over the current Graph screen. To display only the picture, use the Y= Editor to deselect any other functions before opening the graph picture.

From the Graph screen:

1. Press \(\text{F1}\) and select 1:Open.

2. Select the type (Picture), folder, and variable that contain the graph picture you want to open.

   **Note:** If a variable name is not shown on the dialog box, there are no graph pictures in the folder.

3. Press \(\text{ENTER}\).

A graph picture is a drawing object. You cannot trace any curve on a picture.

For Pictures Saved from a Portion of the Graph Screen

When you press \(\text{F1}\) and select 1:Open, the picture is superimposed starting at the upper-left corner of the Graph screen. If the picture was saved from a portion of the Graph screen, it may appear shifted from the underlying graph.

You can specify which screen pixel to use as the upper-left corner.
Deleting a Graph Picture

Unwanted Picture variables take up calculator memory. To delete a variable, use the VAR-LINK screen (2nd [VAR-LINK]) as described in Memory and Variable Management.

From a Program or the Home Screen

To save (store) and open (recall) a graph picture, use the StoPic, RclPic, AndPic, XorPic, and RplcPic commands as described in the Technical Reference module.

To display a series of graph pictures as an animation, use the CyclePic command. For an example, refer to CyclePic Command.

Animating a Series of Graph Pictures

As described earlier in this module, you can save a picture of a graph. By using the CyclePic command, you can flip through a series of graph pictures to create an animation.

CyclePic Command

Before using CyclePic, you must have a series of graph pictures that have the same base name and are sequentially numbered starting with 1 (such as pic1, pic2, pic3, . . . ).

To cycle the pictures, use the syntax:
**Example**

This example program (named **cyc**) generates 10 views of a 3D graph, with each view rotated $10^\circ$ further around the Z axis. For information about each command, refer to the
### Technical Reference

Module. For information about using the Program Editor, refer to Programming.

<table>
<thead>
<tr>
<th>Program Listing</th>
<th>Every Other Graph from Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>:cyc()</td>
<td><img src="image1" alt="Graph 1" /></td>
</tr>
<tr>
<td>:Prgm</td>
<td><img src="image2" alt="Graph 2" /></td>
</tr>
<tr>
<td>:local I</td>
<td><img src="image3" alt="Graph 3" /></td>
</tr>
<tr>
<td>:setMode(“graph”,”3d”)</td>
<td><img src="image4" alt="Graph 4" /></td>
</tr>
<tr>
<td>:70→eyeθ</td>
<td><img src="image5" alt="Graph 5" /></td>
</tr>
<tr>
<td>:-10→xmin</td>
<td><img src="image6" alt="Graph 6" /></td>
</tr>
<tr>
<td>:10→xmax</td>
<td><img src="image7" alt="Graph 7" /></td>
</tr>
<tr>
<td>:14→xgrid</td>
<td><img src="image8" alt="Graph 8" /></td>
</tr>
<tr>
<td>:-10→ymin</td>
<td><img src="image9" alt="Graph 9" /></td>
</tr>
<tr>
<td>:10→ymax</td>
<td><img src="image10" alt="Graph 10" /></td>
</tr>
<tr>
<td>:14→ygrid</td>
<td><img src="image11" alt="Graph 11" /></td>
</tr>
<tr>
<td>:-10→zmin</td>
<td><img src="image12" alt="Graph 12" /></td>
</tr>
<tr>
<td>:10→zmax</td>
<td><img src="image13" alt="Graph 13" /></td>
</tr>
<tr>
<td>:1→zscl</td>
<td><img src="image14" alt="Graph 14" /></td>
</tr>
<tr>
<td>: Define the function</td>
<td><img src="image15" alt="Graph 15" /></td>
</tr>
<tr>
<td>: (x^3<em>y-y^3</em>x)/390→z1(x,y)</td>
<td><img src="image16" alt="Graph 16" /></td>
</tr>
<tr>
<td>: Generate pics and rotate</td>
<td><img src="image17" alt="Graph 17" /></td>
</tr>
<tr>
<td>: For i,1,10,1</td>
<td><img src="image18" alt="Graph 18" /></td>
</tr>
<tr>
<td>: i*10→eyeθ</td>
<td><img src="image19" alt="Graph 19" /></td>
</tr>
<tr>
<td>: DispG</td>
<td><img src="image20" alt="Graph 20" /></td>
</tr>
<tr>
<td>: StoPic #(&quot;pic&quot; &amp; string(i))</td>
<td><img src="image21" alt="Graph 21" /></td>
</tr>
<tr>
<td>: EndFor</td>
<td><img src="image22" alt="Graph 22" /></td>
</tr>
<tr>
<td>: Display animation</td>
<td><img src="image23" alt="Graph 23" /></td>
</tr>
<tr>
<td>: CyclePic &quot;pic&quot;,10,.5,5,-1</td>
<td><img src="image24" alt="Graph 24" /></td>
</tr>
<tr>
<td>: EndPrgm</td>
<td><img src="image25" alt="Graph 25" /></td>
</tr>
</tbody>
</table>
Additional Graphing Topics

Comments start with ♣. Press:

Note: Due to its complexity, this program takes several minutes to run.

After entering this program on the Program Editor, go to the Home screen and enter \texttt{cyc( )}.

**Saving and Opening a Graph Database**

A graph database is the set of all elements that define a particular graph. By saving a graph database as a GDB variable, you can recreate that graph at a later time by opening its stored database variable.

**Elements in a Graph Database**

A graph database consists of:

- Mode settings (\texttt{(MODE)}) for \texttt{Graph}, \texttt{Angle}, \texttt{Complex Format}, and \texttt{Split Screen} (only if you are using the two-graph mode).
- All functions in the \texttt{Y= Editor} (\texttt{[Y=]}), including display styles and which functions are selected.
- Table parameters (\texttt{[TABLE]}), Window variables (\texttt{[WINDOW]}), and graph formats: \texttt{F1 9}
  - or -
  \texttt{[1]}

---

Addendum: This section provides additional information about graphing topics, which may include detailed instructions, examples, or advanced concepts related to creating, saving, and opening graph databases. The text emphasizes the importance of understanding the provided program and its implications for graphing operations. It also highlights the complexity of the program and the time required for execution, prompting users to be patient and prepared for a comprehensive experience with graphing. The description of elements in a graph database offers a thorough insight into the structure and components that make up a graph database, guiding users on the significance of each component and how they interact within the graphing interface.
A graph database does not include drawn objects or stat plots.

Note: In two-graph mode, the elements for both graphs are saved in a single database.

**Saving the Current Graph Database**

From the Y= Editor, Window Editor, Table screen, or Graph screen:

1. Press `F1` and select **2:Save Copy As**.
2. Specify the folder and a unique variable name.
3. Press **ENTER**. After typing in an input box such as Variable, you must press **ENTER** twice.

Note: If you start from the Graph screen, be sure to use Type=GDB.

**Opening a Graph Database**

Caution: When you open a graph database, all information in the current database is replaced. You may want to store the current graph database before opening a stored database.
From the Y= Editor, Window Editor, Table screen, or Graph screen:

1. Press \( \text{F1} \) and select **1:Open**.

2. Select the folder and variable that contain the graph database you want to open.

3. Press **ENTER**.

**Note:** If you start from the Graph screen, be sure to use Type=GDB.

**Deleting a Graph Database**

Unused GDB variables take up calculator memory. To delete them, use the VAR-LINK screen (\( \text{2nd} [\text{VAR-LINK}] \)) described in *Memory and Variable Management*.

**From a Program or the Home Screen**

You can save (store) and open (recall) a graph database by using the **StoGDB** and **RclGDB** commands as described in the *Technical Reference* module.
Split Screens

Setting and Exiting the Split Screen Mode

To set up a split screen, use the MODE dialog box to specify the applicable mode settings. After you set up the split screen, it remains in effect until you change it.

Setting the Split Screen Mode

1. Press \texttt{MODE} to display the \texttt{MODE} dialog box.

2. Because the modes related to split screens are listed on the second page of the \texttt{MODE} dialog box, either:
   - Use \texttt{D} to scroll down.
   — or —
   - Press \texttt{F2} to display \texttt{Page 2}.

3. Set the \texttt{Split Screen} mode to either of the following settings. For the procedure used to change a mode setting, refer to \textit{Operating the Calculator}.

\begin{center}
\begin{tabular}{l}
\textbf{Split Screen Settings} \\
\hline
\texttt{TOP-BOTTOM} \\
\hline
\texttt{LEFT-RIGHT} \\
\hline
\end{tabular}
\end{center}
Setting the Initial Applications

Before pressing \[\text{ENTER}\] to close the MODE dialog box, you can use the **Split 1 App** and **Split 2 App** modes to select the applications you want to use.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Specifies the application in the:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split 1 App</td>
<td>Top or left part of the split screen.</td>
</tr>
<tr>
<td>Split 2 App</td>
<td>Bottom or right part of the split screen.</td>
</tr>
</tbody>
</table>

If you set **Split 1 App** and **Split 2 App** to the same application, the calculator exits the split screen mode and displays the application full screen.

You can open different applications after the split screen is displayed.

**Note:** In two-graph mode, described in *Additional Graphing Topics*, the same application can be in both parts of a split screen.
Other Modes that Affect a Split Screen

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Graphs</td>
<td>Lets you set up and display two independent sets of graphs.</td>
</tr>
<tr>
<td>Note: Leave this set to 1 unless you have read the applicable section in Additional Graphing Topics.</td>
<td></td>
</tr>
</tbody>
</table>

Split Screens and Pixel Coordinates

The calculator has commands that use pixel coordinates to draw lines, circles, etc., on the Graph screen. The following charts show how the Split Screen and Split Screen Ratio mode settings affect the number of pixels available on the Graph screen.

Note:

- For a list of drawing commands, refer to “Drawing on the Graph Screen” in Programming.
- Due to the border that indicates the active application, split screens have a smaller displayable area than a full screen.

TI-89 Titanium:

<table>
<thead>
<tr>
<th>Split</th>
<th>Ratio</th>
<th>Split 1 App</th>
<th>Split 2 App</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>FULL</td>
<td>N/A</td>
<td>0 – 158</td>
<td>0 – 76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
### Split Screens

#### Voyage™ 200:

<table>
<thead>
<tr>
<th>Split</th>
<th>Ratio</th>
<th>x</th>
<th>y</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOP–BOTTOM</td>
<td>1:1</td>
<td>0–154</td>
<td>0–34</td>
<td>0–154</td>
<td>0–34</td>
</tr>
<tr>
<td>LEFT–RIGHT</td>
<td>1:1</td>
<td>0–76</td>
<td>0–72</td>
<td>0–76</td>
<td>0–72</td>
</tr>
</tbody>
</table>

#### Exiting the Split Screen Mode

**Method 1:** Press **[MODE]** to display the MODE dialog box. Then set **Split Screen = FULL**. When you press **[ENTER]** to close the dialog box, the full-sized screen shows the application specified in **Split 1 App**.

---

**Split Screens**
When You Turn Off the Calculator

Turning the calculator off does not exit the split screen mode.

<table>
<thead>
<tr>
<th>If the calculator is turned off:</th>
<th>When you turn the calculator on again:</th>
</tr>
</thead>
<tbody>
<tr>
<td>When you press [2nd] [OFF]</td>
<td>The split screen is still in effect, but the Home screen is always displayed in place of the application that was active when you pressed [2nd] [OFF].</td>
</tr>
</tbody>
</table>

By the Automatic Power Down™ (APD™) feature, or when you press [Home] [OFF]. The split screen is just as you left it.

Split-Screen Status Indicators on the Apps Desktop

To return to the Apps desktop, press [Apps]. The split-screen status appears at the top of the Apps desktop with the names of the open Apps and the portions of the screen in which each App is displayed.

Note: The Apps desktop always appears in the full-screen view.
<table>
<thead>
<tr>
<th>Split screen indicator</th>
<th>Description</th>
</tr>
</thead>
</table>
| ![Top-bottom split screen](image) | Top-bottom split screen  
  - 1 indicates the application that will appear in the top portion of the screen.  
  - 2 indicates the application that will appear in the bottom portion of the screen.  
  The highlighted numeral indicates the active portion of the split screen. |
| ![Left-right split screen](image) | Left-right split screen  
  - 1 indicates the application that will appear in the left portion of the screen.  
  - 2 indicates the application that will appear in the right portion of the screen.  
  The highlighted numeral indicates the active portion of the split screen. |
Selecting the Active Application

With a split screen, only one of the two applications can be active at a time. You can easily switch between existing applications, or you can open a different application.

The Active Application

- The active application is indicated by a thick border.
- The toolbar and status line, which are always the full width of the display, are associated with the active application.
- For applications that have an entry line (such as the Home screen and Y= Editor), the entry line is the full width of the display only when that application is active.

Switching between Applications

Press \[ \text{2nd} \ [ \text{+} ] \] (second function of \[ \text{APPS} \]) to switch from one application to the other.
Opening a Different Application

Method 1:  1. Use 2nd [+] to switch to the application you want to replace.
   2. Use APPS or § (such as § [WINDOW]) to select the new application.

If you select an application that is already displayed, the calculator switches to that application.

Method 2:  3. Press MODE and then F2.
   4. Change Split 1 App and/or Split 2 App.

If you set Split 1 App and Split 2 App to the same application, the calculator exits the split screen mode and displays the application full screen.

Note: In two-graph mode, described in Additional Graphing Topics, the same application can be in both parts of a split screen.
Using 2nd QUIT to Display the Home Screen

Note: Pressing \textbf{[2nd] [QUIT]} twice always exits the split screen mode.

<table>
<thead>
<tr>
<th>If the Home screen:</th>
<th>Pressing \textbf{[2nd] [QUIT]}:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is not already displayed</td>
<td>Opens the Home screen in place of the active application.</td>
</tr>
<tr>
<td>Is displayed, but is not the active application</td>
<td>Switches to the Home screen and makes it the active application.</td>
</tr>
<tr>
<td>Is the active application</td>
<td>Exits the split screen mode and displays a full-sized Home screen.</td>
</tr>
</tbody>
</table>

\textbf{When Using a Top-Bottom Split}

When you select a TOP-BOTTOM split, remember that the entry line and the toolbar are always associated with the active application. For example:

Entry line is for the active \textit{Y= Editor}, \textit{not} the Graph screen.

Toolbar is for the active Graph screen, \textit{not} the \textit{Y= Editor}. 

Split Screens
Note: Both Top-Bottom and Left-Right splits use the same methods to select an application.
Overview of List, Data, and Matrix Variables

To use the Data/Matrix Editor effectively, you must understand list, data, and matrix variables.

**List Variable**

A list is a series of items (numbers, expressions, or character strings) that may or may not be related. Each item is called an element. In the Data/Matrix Editor, a list variable:

- Is shown as a single column of elements, each in a separate cell.
- Must be continuous; blank or empty cells are not allowed within the list.
- Can have up to 999 elements.

1. Column title and header cells are not saved as part of the list.

If you enter more than one column of elements in a list variable, it is converted automatically into a data variable.
On the Home screen (or anywhere else you can use a list), you can enter a list as a series of elements enclosed in braces { } and separated by commas.

Although you must use commas to separate elements on the entry line, spaces separate the elements in the history area.

To refer to a specified element in a list, use the format shown to the right.

1 Name of list variable
2 Element number (or index number)

**Note:** After creating a list in the Data/Matrix Editor, you can use the list in any application (such as the Home screen).

**Data Variable**

A data variable is essentially a collection of lists that may or may not be related. In the Data/Matrix Editor, a data variable:

- Can have up to 99 columns.
- Can have up to 999 elements in each column. Depending on the kind of data, all columns may not have to be the same length.
- Must have continuous columns; blank or empty cells are not allowed within a column.
**Note:** For stat calculations, columns must have the same length.

From the Home screen or a program, you can use the `NewData` command to create a data variable that consists of existing lists.

1. Name of data variable to create  
2. Names of existing list

Although you cannot directly display a data variable on the Home screen, you can display a specified column or element.

1. Name of data variable  
2. Column number  
3. Column number  
4. Element number in the column

For example:

1. Displays column 1 of the variable `data1`.  
2. Displays element 1 in column 1 of the variable `data1`. 
**Matrix Variable**

A matrix is a rectangular array of elements. When you create a matrix in the Data/Matrix Editor, you must specify the number of rows and columns (although you can add or delete rows and columns later). In the Data/Matrix Editor, a matrix variable:

- Looks similar to a data variable, but all columns must have the same length.
- Is initially created with 0 in each cell. You can then enter the applicable value in place of 0.

From the Home screen or a program, you can use `STO>` to store a matrix with either of the equivalent methods shown to the right.

1. row 1
2. row 2
3. row 1
4. row 2

Although you enter the matrix as shown above, it is pretty printed in the history area in traditional matrix form.

After creating a matrix in the Data/Matrix Editor, you can use the matrix in any application (such as the Home screen).

**Note:** Use brackets to refer to a specific element in a matrix. For example, enter `mat1[2,1]` to access the 1st element in the 2nd row.
Starting a Data/Matrix Editor Session

Each time you start the Data/Matrix Editor, you can create a new variable, resume using the current variable (the variable that was displayed the last time you used the Data/Matrix Editor), or open an existing variable.

Creating a New Data, Matrix, or List Variable

1. Press \textbf{APPs} and then select the Data/Matrix icon. Press \textbf{ENTER}.

2. Select \textbf{3:New}.

3. Specify the applicable information for the new variable.

<table>
<thead>
<tr>
<th>Item</th>
<th>Lets you:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Select the type of variable to create. Press $\leftarrow$ to display a menu of available types.</td>
</tr>
<tr>
<td>Folder</td>
<td>Select the folder in which the new variable will be stored. Press $\leftarrow$ to display a menu of existing folders. For information about folders, refer to the Calculator Home Screen module.</td>
</tr>
<tr>
<td>Item</td>
<td>Lets you:</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Variable</td>
<td>Type a new variable name. If you specify a variable that already exists, an error message will be displayed when you press [ENTER]. When you press [ESC] or [ENTER] to acknowledge the error, the NEW dialog box is redisplayed.</td>
</tr>
<tr>
<td>Row dimension and</td>
<td>If Type = Matrix, type the number of rows and columns in the matrix.</td>
</tr>
<tr>
<td>Col dimension</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If you do not type a variable name, your calculator displays the Home screen.

4. Press [ENTER] (after typing in an input box such as **Variable**, press [ENTER] twice) to create and display an empty variable in the Data/Matrix Editor.

**Using the Current Variable**

You can leave the Data/Matrix Editor and go to another application at any time. To return to the variable that was displayed when you left the Data/Matrix Editor, launch Data/Matrix Editor again and select 1:**Current**.
Creating a New Variable from the Data/Matrix Editor

From the Data/Matrix Editor:

1. Press $\text{F1}$ and select 3:New.
2. Specify the type, folder, and variable name. For a matrix, also specify the number of rows and columns.

Opening Another Variable

You can open another variable at any time.

1. From the Data/Matrix Editor, press $\text{F1}$ and select 1:Open.
   – or –
   From any application, launch Data/Matrix Editor again and select 2:Open.
2. Select the type, folder, and variable to open.
3. Press $\text{ENTER}$.

Note: Variable shows the first existing variable in alphabetic order. If there are no existing variables, nothing is displayed.
Deleting a Variable

Because all Data/Matrix Editor variables are saved automatically, you can accumulate quite a few variables, which take up memory.

To delete a variable, use the VAR-LINK screen (\texttt{2nd \texttt{[VAR-LINK]}}). For information about VAR-LINK, refer to \textit{Memory and Variable Management}.

Entering and Viewing Cell Values

If you create a new variable, the Data/Matrix Editor is initially blank (for a list or data variable) or filled with zeros (for a matrix). If you open an existing variable, the values in that variable are displayed. You can then enter additional values or edit the existing ones.

The Data/Matrix Editor Screen

A blank Data/Matrix Editor screen is shown below. When the screen is displayed initially, the cursor highlights the cell at row 1, column 1.

- Variable type
- Column headers
- Row numbers
- Row and column number of highlighted cell
- Column title cells, used to type a title for each column
When values are entered, the entry line shows the full value of the highlighted cell.

**Note:** Use the title cell at the very top of each column to identify the information in that column.

### Entering or Editing a Value in a Cell

You can enter any type of expression in a cell (number, variable, function, string, etc.).

1. Move the cursor to highlight the cell you want to enter or edit.
2. Press **ENTER** or **F3** to move the cursor to the entry line.
3. Type a new value or edit the existing one.
4. Press **ENTER** to enter the value into the highlighted cell.

When you press **ENTER**, the cursor automatically moves to highlight the next cell so that you can continue entering or editing values. However, the variable type affects the direction that the cursor moves.

**Note:** To enter a new value, you can start typing without pressing **ENTER** or **F3** first. However, you must use **ENTER** or **F3** to edit an existing value.

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>After pressing <strong>ENTER</strong>, the cursor moves:</th>
</tr>
</thead>
<tbody>
<tr>
<td>List or data</td>
<td>Down to the cell in the next row.</td>
</tr>
<tr>
<td>Matrix</td>
<td>Right to the cell in the next column. From the last cell in a row, the cursor automatically moves to the first cell in the next row. This lets you enter values for row1, row2, etc.</td>
</tr>
</tbody>
</table>
Scrolling through the Editor

<table>
<thead>
<tr>
<th>To move the cursor:</th>
<th>Press:</th>
</tr>
</thead>
<tbody>
<tr>
<td>One cell at a time</td>
<td>≦, ≧, ≫, or ≪</td>
</tr>
<tr>
<td>One page at a time</td>
<td>2nd and then ≦, ≧, ≫, or ≪</td>
</tr>
<tr>
<td>Go to row 1 in the current column or to the last row that contains data for any column on the screen, respectively. If the cursor is in or past that last row, ≦ ≦ goes to row 999.</td>
<td>≦ ≦ or ≦ ≦</td>
</tr>
<tr>
<td>Go to column 1 or to the last column that contains data, respectively. If the cursor is in or past that last column, ≦ ≫ goes to column 99.</td>
<td>≦ ≫ or ≦ ≫</td>
</tr>
</tbody>
</table>

Note: To enter a value from the entry line, you can also use ≦ or ≦.

When you scroll down/up, the header row remains at the top of the screen so that the column numbers are always visible. When you scroll right/left, the row numbers remain on the left side of the screen so that they are always visible.

How Rows and Columns Are Filled Automatically

When you enter a value in a cell, the cursor moves to the next cell. However, you can move the cursor to any cell and enter a value. If you leave gaps between cells, your device handles the gaps automatically.
In a list variable, a cell in the gap is undefined until you enter a value for the cell.

Note: If you enter more than one column of elements in a list variable, it is converted automatically into a data variable.

In a data variable, gaps in a column are handled the same as a list. However, if you leave a gap between columns, that column is blank.

Note: Although you specify the size of a matrix when you create it, you can easily add additional rows and/or columns.
Changing the Cell Width

The cell width affects how many characters are displayed in any cell. To change the cell width in the Data/Matrix Editor:

1. To display the FORMATS dialog box, press:
   \[ F1 \] \[ 9 \]
   – or –
   \[ \downarrow \] \[ \rightarrow \]

   ![FORMATS dialog box]

   Cell width is the maximum number of characters that can be displayed in a cell. All cells have the same cell width.

   **Note**: Remember, to see a number in full precision, you can always highlight the cell and look at the entry line.

2. With the current **Cell Width** setting highlighted, press \( \uparrow \) or \( \downarrow \) to display a menu of digits (3 through 12).

3. Move the cursor to highlight a number and press [ENTER]. (For single-digit numbers, you can type the number and press [ENTER].)

4. Press [ENTER] to close the dialog box.
Clearing a Column or all Columns

This procedure erases the contents of a column. It does not delete the column.

To clear: | Do this:
---|---
A column | 1. Move the cursor to any cell in the column.
       | 2. Press: `2nd [F6]` and select **5:Clear Column**.
       | (This item is not available for a matrix.)
All columns | Press `F1` and select **8:Clear Editor**. When prompted for confirmation, press `ENTER` (or `ESC` to cancel).

**Note:** For a list or data variable, a clear column is empty. For a matrix, a clear column contains zeros.

Defining a Column Header with an Expression

For a list variable or a column in a data variable, you can enter a function in the column header that automatically generates a list of elements. In a data variable, you can also define one column in terms of another.

**Entering a Header Definition**

In the Data/Matrix Editor:
1. Move the cursor to any cell in the column and press [F4].
   – or –
   Move the cursor to the header cell (c1, c2, etc.) and press [ENTER].

   **Notes:**
   • [ENTER] is not required if you want to type a new definition or replace the existing one. However, if you want to edit the existing definition, you must press [ENTER].
   • To view an existing definition, press [F4] or move the cursor to the header cell and look at the entry line.

2. Type the new expression, which replaces any existing definition.

   If you used [F4] or [ENTER] in Step 1, the cursor moved to the entry line and highlighted the existing definition, if any. You can also:
   • Press [CLEAR] to clear the highlighted expression. Then type the new expression.
   – or –
• Press ◁ or ▶ to remove the highlighting. Then edit the old expression.

**Note:** To cancel any changes, press [ESC] before pressing [ENTER].

### You can use an expression that:

<table>
<thead>
<tr>
<th>Generates a series of numbers.</th>
<th>For example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1=seq(x^2,x,1,5)</td>
<td>c1={1,2,3,4,5}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Refers to another column.</th>
<th>c2=2*c1</th>
</tr>
</thead>
<tbody>
<tr>
<td>c4=c1*c2–sin(c3)</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The seq function is described in the Technical Reference module. If you refer to an empty column, you will get an error message unless **Auto-calculate = OFF**.

3. Press [ENTER], ◁, or ▶ to save the definition and update the columns.

₁ c1=seq(x,x,1,7); ₂ c2=2*c1; ₃ You cannot directly change a locked cell (₁) since it is defined by the column header.

**Note:** For a data variable, header definitions are saved when you leave the Data/ Matrix Editor. For a list variable, the definitions are not saved (only their resulting cell values).
Clearing a Header Definition

1. Move the cursor to any cell in the column and press $\text{F4}$.
   – or –
   Move the cursor to the header cell ($c_1$, $c_2$, etc.) and press $\text{ENTER}$.
2. Press $\text{CLEAR}$ to clear the highlighted expression.
3. Press $\text{ENTER}$, $\text{⌫}$, or $\text{→}$.

Using an Existing List as a Column

Suppose you have one or more existing lists, and you want to use those existing lists as columns in a data variable.

<table>
<thead>
<tr>
<th>From the:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data/Matrix Editor</td>
<td>In the applicable column, use $\text{F4}$ to define the column header. Refer to the existing list variable. For example: $c_1=\text{list1}$</td>
</tr>
<tr>
<td>Home screen or a program</td>
<td>Use the $\text{NewData}$ command as described in the <em>Technical Reference</em> module. For example:</td>
</tr>
<tr>
<td></td>
<td>$\text{NewData datavar, list1 [, list2] [, list3] ...}$</td>
</tr>
</tbody>
</table>

1 Data variable. If this data variable already exists, it will be redefined based on the specified lists.
2 Existing list variables to copy to columns in the data variable.
**Note:** If you have a CBL 2™ or CBR™, use these techniques for your collected lists. Use 2nd [VAR-LINK] to see existing list variables.

**To Fill a Matrix with a List**

You cannot use the Data/Matrix Editor to fill a matrix with a list. However, you can use the list►mat command from the Home screen or a program. For information, refer to the Technical Reference module.

**The Auto-calculate Feature**

For list and data variables, the Data/Matrix Editor has an Auto-calculate feature. By default, Auto-calculate = ON. Therefore, if you make a change that affects a header definition (or any column referenced in a header definition), all header definitions are recalculated automatically. For example:

- If you change a header definition, the new definition is applied automatically.
- If column 2’s header is defined as c2=2*c1, any change you make in column 1 is automatically reflected in column 2.

To turn Auto-calculate off and on from the Data/Matrix Editor:

1. Press:  
   
   - or -

2. [▲ ▼] Change **Auto-Calculate** to OFF or ON.

3. Press **ENTER** to close the dialog box.
If Auto-calculate = OFF and you make changes as described above, the header definitions are not recalculated until you set Auto-calculate = ON.

**Note:** You may want to set Auto-calculate = OFF to make changes without recalculating each time, enter a definition such as c1=c2+c3 before you enter columns 2 and 3, or override any errors in a definition until you can debug the error.

### Using Shift and CumSum Functions in a Column Header

When defining a column header, you can use the shift and cumSum functions as described below. These descriptions differ slightly from the Technical Reference module. This section describes how to use the functions in the Data/Matrix Editor. The Technical Reference module gives a more general description for the Home screen or a program.

### Using the Shift Function

The shift function copies a base column and shifts it up or down by a specified number of elements. Use [F4] to define a column header with the syntax:

\[
\text{shift (column [,integer])}
\]

1. Column used as the base for the shift.
2. Number of elements to shift (positive shifts up; negative shifts down).
   Default is -1.

For example, for a two-element shift up and down:
Using the CumSum Function

The **cumSum** function returns a cumulative sum of the elements in a base column. Use `F4` to define a column header with the syntax:

\[ \text{cumSum} (\text{column}) \]

Column used as the base for the cumulative sum.

For example:

\[
\begin{array}{c|c|c}
\text{c1} & \text{c2} & \\
1 & 1 & 1+2 \\
2 & 3 & 1+2+3+4 \\
3 & 6 & \\
4 & 10 & \\
\end{array}
\]

**Note**: To enter **cumSum**, type it, select it from the CATALOG, or press `2nd [MATH]` and select it from the List submenu.
Sorting Columns

After entering information in a data, list, or matrix variable, you can easily sort a specified column in numeric or alphabetical order. You can also sort all columns as a whole, based on a “key” column.

Sorting a Single Column

In the Data/Matrix Editor:

1. Move the cursor to any cell in the column.
2. Press: \(2\text{nd} \ [F6]\) and select \textbf{3:Sort Column}.

Numbers are sorted in ascending order.
Character strings are sorted in alphabetical order.
Sorting All Columns Based on a “Key” Column

Consider a database structure in which each column along the same row contains related information (such as a student’s first name, last name, and test scores). In such a case, sorting only a single column would destroy the relationship between the columns.

In the Data/Matrix Editor:

1. Move the cursor to any cell in the “key” column.

2. In this example, move the cursor to the second column (c2) to sort by last name.

   Note: For a list variable, this is the same as sorting a single column.


   Note: This menu item is not available if any column is locked.

When using this procedure for a data variable:

- All columns must have the same length.
- None of the columns can be locked (defined by a function in the column header). When the cursor is in a locked column, \(\mathord{\text{ }}\) is shown at the beginning of the entry line.
Saving a Copy of a List, Data, or Matrix Variable

You can save a copy of a list, data, or matrix variable. You can also copy a list to a data variable, or you can select a column from a data variable and copy that column to a list.

Valid Copy Types

<table>
<thead>
<tr>
<th>You can copy a:</th>
<th>To a:</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td>List or data</td>
</tr>
<tr>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>Data column</td>
<td>List</td>
</tr>
<tr>
<td>Matrix</td>
<td>Matrix</td>
</tr>
</tbody>
</table>

Note: A list is automatically converted to a data variable if you enter more than one column of information.

Procedure

From the Data/Matrix Editor:

1. Display the variable that you want to copy.
2. Press \[ F1 \] and select \textbf{2:Save Copy As}. 

3. In the dialog box:
   - Select the \textbf{Type} and \textbf{Folder} for the copy.
   - Type a variable name for the copy.
   - When available, select the column to copy from.

   \textbf{Note:} If you type the name of an existing variable, its contents will be replaced.

   ① Column is dimmed unless you copy a data column to a list. The column information is not used for other types of copies.

4. Press \[ \text{ENTER} \] (after typing in an input box such as Variable, you must press \[ \text{ENTER} \] twice).

\textbf{To Copy a Data Column to a List}

A data variable can have multiple columns, but a list variable can have only one column. Therefore, when copying from a data variable to a list, you must select the column that you want to copy.

① List variable to copy to.

② Data column that will be copied to the list. By default, this shows the column that contains the cursor.
Overview of Steps in Statistical Analysis

This section gives an overview of the steps used to perform a statistical calculation or graph a statistical plot. For detailed descriptions, refer to the following pages.

1. Set Graph mode (MODE) to FUNCTION.

2. Enter stat data in the Data/Matrix Editor.
   
   **Note:** Refer to the Data/Matrix Editor module for details on entering data in the Data/Matrix Editor.

3. Perform stat calculations to find stat variables or fit data to a model (F5).

4. Define and select stat plots (F2 and then F1).
   
   **Note:** You can also use the Y= Editor to define and select stat plots and y(x) functions.

5. Define the viewing window (VIEW [WINDOW]).
6. Change the graph format if necessary.

7. **F1 9**
   — or —
   ![Graph](image)
   Graph the selected equations (♦ [GRAPH]).

---

**Performing a Statistical Calculation**

From the Data/Matrix Editor, use the **F5 Calc** toolbar menu to perform statistical calculations. You can analyze one-variable or two-variable statistics, or perform several types of regression analyses.

**The Calculate Dialog Box**

You must have a data variable opened. The Data/Matrix Editor will not perform statistical calculations with a list or matrix variable.
From the Data/Matrix Editor:

1. Press F5 to display the **Calculate** dialog box.

This example shows all items as active. On your calculator, items are active only if they are valid for the current settings of **Calculation Type** and **Freq and Categories**.

**Note:** If an item is not valid for the current settings, it will appear dimmed. You cannot move the cursor to a dimmed item.

![Pathname of the data variable]

![Main/Build Calculate dialog box]
2. Specify applicable settings for the active items.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculation Type</td>
<td>Select the type of calculation.</td>
</tr>
<tr>
<td>x</td>
<td>Type the column number in the Data/Matrix Editor (C1, C2, etc.) used for x values, the independent variable.</td>
</tr>
<tr>
<td>Y</td>
<td>Type the column number used for y values, the dependent variable. This is required for all Calculation Types except OneVar.</td>
</tr>
<tr>
<td>Store RegEQ to</td>
<td>If Calculation Type is a regression analysis, you can select a function name (y1(x), y2(x), etc.). This lets you store the regression equation so that it will be displayed in the Y= Editor.</td>
</tr>
<tr>
<td>Use Freq and Categories?</td>
<td>Select NO or YES. Note that Freq, Category, and Include Categories are active only when Use Freq and Categories? = YES.</td>
</tr>
<tr>
<td>Freq</td>
<td>Type the column number that contains a “weight” value for each data point. If you do not enter a column number, all data points are assumed to have the same weight (1).</td>
</tr>
<tr>
<td>Category</td>
<td>Type the column number that contains a category value for each data point.</td>
</tr>
<tr>
<td>Include Categories</td>
<td>If you specify a Category column, you can use this item to limit the calculation to specified category values. For example, if you specify {1,4}, the calculation uses only data points with a category value of 1 or 4.</td>
</tr>
</tbody>
</table>
Note: To use an existing list variable for x, y, Freq, or Category, type the list name instead of a column number. An example using Freq, Category, and Include Categories is available.


The results are displayed on the STAT VARS screen. The format depends on the Calculation Type. For example:

<table>
<thead>
<tr>
<th>Calculation Type = OneVar</th>
<th>Calculation Type = LinReg</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STAT VARS</strong></td>
<td><strong>STAT VARS</strong></td>
</tr>
<tr>
<td>( \bar{x} = 33.428571 )</td>
<td>( y = a \cdot x )</td>
</tr>
<tr>
<td>( \bar{x} = 234. )</td>
<td>( a = 0.081561 )</td>
</tr>
<tr>
<td>( \sum x = 11576 )</td>
<td>( b = 12.012431 )</td>
</tr>
<tr>
<td>( \Sigma x^2 = 25.012329 )</td>
<td>( \text{corr} = 0.557317 )</td>
</tr>
<tr>
<td>( n = 7 )</td>
<td>( r^2 = 0.316457 )</td>
</tr>
<tr>
<td>( \min = 4 )</td>
<td></td>
</tr>
<tr>
<td>( \max = 9 )</td>
<td></td>
</tr>
<tr>
<td>( \text{med} = 5.1 )</td>
<td></td>
</tr>
</tbody>
</table>

When \( \downarrow \) is shown instead of \( = \), you can scroll for additional results.

Note: Any undefined data points (shown as \text{undef}) are ignored in a stat calculation.

4. To close the STAT VARS screen, press [ENTER].

Redisplaying the STAT VARS Screen

The Data/Matrix Editor’s Stat toolbar menu redisplays the previous calculation results (until they are cleared from memory).

\( \mathbb{D} \) \( 2 \text{nd} \) [F7]
Previous results are cleared when you:

- Edit the data points or change the Calculation Type.
- Open another data variable or reopen the same data variable (if the calculation referred to a column in a data variable). Results are also cleared if you leave and then reopen the Data/Matrix Editor with a data variable.
- Change the current folder (if the calculation referred to a list variable in the previous folder).

**Statistical Calculation Types**

As described in the previous section, the Calculate dialog box lets you specify the statistical calculation you want to perform. This section gives more information about the calculation types.
Selecting the Calculation Type

From the Calculate dialog box ([F5]), highlight the current setting for the Calculation Type and press [Enter].

You can then select from a menu of available types.

<table>
<thead>
<tr>
<th>Calc Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OneVar</td>
<td>One-variable statistics — Calculates the statistical variables.</td>
</tr>
<tr>
<td>TwoVar</td>
<td>Two-variable statistics — Calculates the statistical variables.</td>
</tr>
<tr>
<td>CubicReg</td>
<td>Cubic regression — Fits the data to the third-order polynomial ( y=ax^3+bx^2+cx+d ). You must have at least four data points.</td>
</tr>
<tr>
<td></td>
<td>• For four points, the equation is a polynomial fit.</td>
</tr>
<tr>
<td></td>
<td>• For five or more points, it is a polynomial regression.</td>
</tr>
<tr>
<td>ExpReg</td>
<td>Exponential regression — Fits the data to the model equation ( y=ab^x ) (where a is the y-intercept) using a least-squares fit and transformed values ( x ) and ( \ln(y) ).</td>
</tr>
<tr>
<td>Calc Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>LinReg</td>
<td>Linear regression — Fits the data to the model $y=ax+b$ (where $a$ is the slope, and $b$ is the y-intercept) using a least-squares fit and $x$ and $y$.</td>
</tr>
<tr>
<td>LnReg</td>
<td>Logarithmic regression — Fits the data to the model equation $y=a+b \ln(x)$ using a least-squares fit and transformed values $\ln(x)$ and $y$.</td>
</tr>
<tr>
<td>Logistic</td>
<td>Logistic regression — Fits the data to the model $y=a/(1+b<em>e^{(c</em>x)})+d$ and updates all the system statistics variables.</td>
</tr>
<tr>
<td>MedMed</td>
<td>Median-Median — Fits the data to the model $y=ax+b$ (where $a$ is the slope, and $b$ is the y-intercept) using the median-median line, which is part of the resistant line technique. Summary points $\text{medx1, medy1, medx2, medy2, medx3,}$ and $\text{medy3}$ are calculated and stored to variables, but they are not displayed on the STAT VARS screen.</td>
</tr>
<tr>
<td>PowerReg</td>
<td>Power regression — Fits the data to the model equation $y=ax^b$ using a least-squares fit and transformed values $\ln(x)$ and $\ln(y)$.</td>
</tr>
</tbody>
</table>
| QuadReg   | Quadratic regression — Fits the data to the second-order polynomial $y=ax^2+bx+c$. You must have at least three data points.  
  - For three points, the equation is a polynomial fit.  
  - For four or more points, it is a polynomial regression. |
Note: For TwoVar and all regression calculations, the columns that you specify for x and y (and optionally, Freq or Category) must have the same length.

**From the Home Screen or a Program**

Use the applicable command for the calculation that you want to perform. The commands have the same name as the corresponding Calculation Type. Refer to the *Technical Reference* module for information about each command.

**Important:** These commands perform a statistical calculation but do not automatically display the results. Use the ShowStat command to show the calculation results.

**Statistical Variables**

Statistical calculation results are stored to variables. To access these variables, type the variable name or use the VAR-LINK screen as described in *Memory and Variable*
Management. All statistical variables are cleared when you edit the data or change the calculation type. Other conditions that clear the variables are listed.

**Calculated Variables**

Statistical variables are stored as system variables. However, `regCoef` and `regeq` are treated as a list and a function variable, respectively.

<table>
<thead>
<tr>
<th></th>
<th>One Var</th>
<th>Two Var</th>
<th>Regressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean of x values</td>
<td>$\bar{x}$</td>
<td>$\bar{x}$</td>
<td></td>
</tr>
<tr>
<td>sum of x values</td>
<td>$\sum x$</td>
<td>$\sum x$</td>
<td></td>
</tr>
<tr>
<td>sum of $x^2$ values</td>
<td>$\sum x^2$</td>
<td>$\sum x^2$</td>
<td></td>
</tr>
<tr>
<td>sample std. deviation of x</td>
<td>$S_x$</td>
<td>$S_x$</td>
<td></td>
</tr>
<tr>
<td>population std. deviation of x</td>
<td>$\sigma_x$</td>
<td>$\sigma_x$</td>
<td></td>
</tr>
<tr>
<td>number of data points</td>
<td>nStat</td>
<td>nStat</td>
<td></td>
</tr>
<tr>
<td>mean of y values</td>
<td>$\bar{y}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sum of y values</td>
<td>$\sum y$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sum of $y^2$ values</td>
<td>$\sum y^2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sample standard deviation of y</td>
<td>$S_y$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>population std. deviation of y</td>
<td>$\sigma_y$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sum of $x \times y$ values</td>
<td>$\sum xy$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>minimum of x values</td>
<td>minX</td>
<td>minX</td>
<td></td>
</tr>
<tr>
<td><strong>Statistics and Data Plots</strong></td>
<td><strong>One Var</strong></td>
<td><strong>Two Var</strong></td>
<td><strong>Regressions</strong></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>maximum of x values</td>
<td>maxX</td>
<td>minX</td>
<td>regEq</td>
</tr>
<tr>
<td>minimum of y values</td>
<td></td>
<td>minY</td>
<td></td>
</tr>
<tr>
<td>maximum of y values</td>
<td></td>
<td>maxY</td>
<td></td>
</tr>
<tr>
<td>1st quartile</td>
<td>q1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>median</td>
<td>medStat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3rd quartile</td>
<td>q3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>regression equation</td>
<td></td>
<td>regEq</td>
<td></td>
</tr>
<tr>
<td>regression coefficients (a, b, c, d, e)</td>
<td>regCoef</td>
<td></td>
<td></td>
</tr>
<tr>
<td>correlation coefficient ††</td>
<td>corr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>coefficient of determination ††</td>
<td>R^2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>summary points (MedMed only) †</td>
<td>medx1, medy1, medx2, medy2, medx3, medy3</td>
<td>medStat</td>
<td></td>
</tr>
</tbody>
</table>

†† **corr** is defined for a linear regression only; **R^2** is defined for all polynomial regressions.

**Note:**

- If **regeq** is $4x + 7$, then **regCoef** is $\{4 \ 7\}$. To access the “$a$” coefficient (the 1st element in the list), use an index such as **regCoef[1]**.
• 1st quartile is the median of points between \texttt{minX} and \texttt{medStat}, and 3rd quartile is the median of points between \texttt{medStat} and \texttt{maxX}.

\section*{Defining a Statistical Plot}

From the Data/Matrix Editor, you can use the entered data to define several types of statistical plots. You can define up to nine plots at a time.

\subsection*{Procedure}

From the \textbf{Data/Matrix Editor}:

1. Press \texttt{F2} to display the \textbf{Plot Setup} screen. Initially, none of the plots are defined.

2. Move the cursor to highlight the plot number that you want to define.

3. Press \texttt{F1} to define the plot.

This example shows all items as active. On your calculator, items are active only if they are valid for the current setting of \textbf{Plot Type} and use \textbf{Freq and Categories}?.

\textbf{Note}: If an item is not valid for the current settings, it will appear dimmed. You cannot move the cursor to a dimmed item.
4. Specify applicable settings for the active items.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plot Type</td>
<td>Select the type of plot.</td>
</tr>
<tr>
<td>Mark</td>
<td>Select the symbol used to plot the data points: Box (□), Cross (x), Plus (+), Square (□), or Dot (•).</td>
</tr>
<tr>
<td>x</td>
<td>Type the column number in the Data/Matrix Editor (C1, C2, etc.) used for x values, the independent variable.</td>
</tr>
<tr>
<td>y</td>
<td>Type the column number used for y values, the dependent variable. This is active only for Plot Type = Scatter or xyline.</td>
</tr>
<tr>
<td>Hist. Bucket Width</td>
<td>Specifies the width of each bar in a histogram.</td>
</tr>
<tr>
<td>Freq and Categories?</td>
<td>Select NO or YES. Note that Freq, Category, and Include Categories are active only when Freq and Categories? = YES. (Freq is active only for Plot Type = Box Plot or Histogram.)</td>
</tr>
<tr>
<td>Freq</td>
<td>Type the column number that contains a “weight” value for each data point. If you do not enter a column number, all data points are assumed to have the same weight (1).</td>
</tr>
<tr>
<td>Category</td>
<td>Type the column number that contains a category value for each data point.</td>
</tr>
<tr>
<td>Include Categories</td>
<td>If you specify a Category, you can use this to limit the calculation to specified category values. For example, if you specify {1,4}, the plot uses only data points with a category value of 1 or 4.</td>
</tr>
</tbody>
</table>

**Note:**
• Plots defined with column numbers always use the last data variable in the Data/Matrix Editor, even if that variable was not used to create the definition.

• To use an existing list variable for x, y, Freq, or Category, type the list name instead of the column number.

• An example using Freq, Category, and Include Categories is available.

5. Press \( \text{ENTER} \) (after typing in an input box, press \( \text{ENTER} \) twice).

The Plot Setup screen is redisplayed.

The plot you just defined is automatically selected for graphing.

Notice the shorthand definition for the plot.

\[
\text{Plot Type} = \text{Scatter} \\
\text{Mark} = \text{Box} \\
\text{Plot 1:} \quad x = c1 \quad y = c2
\]

Note: Any undefined data points (shown as \text{undef}) are ignored in a stat plot.

Selecting or Deselecting a Plot

From Plot Setup, highlight the plot and press \( \text{F4} \) to toggle it on or off. If a stat plot is selected, it remains selected when you:

• Change the graph mode. (Stat plots are not graphed in 3D mode.)
• Execute a Graph command.
• Open a different variable in the Data/Matrix Editor.

**Copying a Plot Definition**

From **Plot Setup**:

1. Highlight the plot and press [F2].
2. Press → and select the plot number that you want to copy to.
3. Press [ENTER].

**Note**: If the original plot was selected (✓), the copy is also selected.

**Clearing a Plot Definition**

From Plot Setup, highlight the plot and press [F3]. To redefine an existing plot, you do not necessarily need to clear it first; you can make changes to the existing definition. To prevent a plot from graphing, you can deselect it.

**Statistical Plot Types**

When you define a plot as described in the previous section, the Plot Setup screen lets you select the plot type. This section gives more information about the available plot types.
Scatter

Data points from x and y are plotted as coordinate pairs. Therefore, the columns or lists that you specify for x and y must be the same length.

- Plotted points are shown with the symbol that you select as the Mark.
- If necessary, you can specify the same column or list for both x and y.

Xyline

This is a scatter plot in which data points are plotted and connected in the order in which they appear in x and y.

You may want to sort all the columns in the Data/Matrix Editor before plotting.

2nd F6 3 or 2nd F6 4
**Box Plot**

This plots one-variable data with respect to the minimum and maximum data points (minX and maxX) in the set.

- A box is defined by its first quartile (Q1), median (Med), and third quartile (Q3).
- Whiskers extend from minX to Q1 and from Q3 to maxX.

- When you select multiple box plots, they are plotted one above the other in the same order as their plot numbers.
- Use NewPlot to show statistical data as a modified box plot.
- Select Mod Box Plot as the Plot Type when you define a plot in the Data/Matrix Editor.

A modified box plot excludes points outside the interval \([Q1-X, Q3+X]\), where \(X\) is defined as 1.5 \((Q3-Q1)\). These points, called outliers, are plotted individually beyond the box plot’s whiskers, using the mark that you select.
**Histogram**

This plots one-variable data as a histogram. The $x$ axis is divided into equal widths called buckets or bars. The height of each bar (its $y$ value) indicates how many data points fall within the bar’s range.

- When defining the plot, you can specify the **Hist. Bucket Width** (default is 1) to set the width of each bar.

- A data point at the edge of a bar is counted in the bar to the right.

- **ZoomData** ([F2] 9 from the Graph screen, Y= Editor, or Window Editor) adjusts $x_{\text{min}}$ and $x_{\text{max}}$ to include all data points, but it does not adjust the $y$ axis.

  - Use [WINDOW] to set $y_{\text{min}} = 0$ and $y_{\text{max}} = \text{the number of data points expected in the tallest bar.}$

\[
\text{Number of bars} = \frac{x_{\text{max}} - x_{\text{min}}}{\text{Hist. Bucket Width}}
\]
• When you trace (F3) a histogram, the screen shows information about the traced bar.

Using the Y= Editor with Stat Plots

The previous sections described how to define and select stat plots from the Data/Matrix Editor. You can also define and select stat plots from the Y= Editor.
Press [Y=] to display the Y= Editor. Initially, the nine stat plots are located “off the top” of the screen, above the y(x) functions. However, the PLOTS indicator provides some information.

For example, PLOTS 23 means that Plots 2 & 3 are selected.

To see the list of stat plots, use [ ] to scroll above the y(x) functions.

If a Plot is highlighted, this shows the data variable that will be used for the plots.
If a Plot is defined, it shows the same shorthand notation as the Plot Setup screen.

From the Y= Editor, you can perform most of the same operations on a stat plot as you can on any other y(x) function.
**Note:** Plots defined with column numbers always use the last data variable in the Data/Matrix Editor, even if that variable was not used to create the definition.

<table>
<thead>
<tr>
<th>To:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit a plot definition</td>
<td>Highlight the plot and press [F3]. You will see the same definition screen that is displayed in the Data/Matrix Editor.</td>
</tr>
<tr>
<td>Select or deselect a plot</td>
<td>Highlight the plot and press [F4].</td>
</tr>
<tr>
<td>Turn all plots and/or functions off</td>
<td>Press [F5] and select the applicable item. You can also use this menu to turn all functions on.</td>
</tr>
</tbody>
</table>

**Note:** You can not use `@2` to set a plot’s display style. However, the plot definition lets you select the mark used for the plot.

**To Graph Plots and Y= Functions**

As necessary, you can select and graph stat plots and y(x) functions at the same time.

**Graphing and Tracing a Defined Stat Plot**

After entering the data points and defining the stat plots, you can graph the selected plots by using the same methods you used to graph a function from the Y= Editor (as described in *Basic Function Graphing*).
Defining the Viewing Window

Stat plots are displayed on the current graph, and they use the Window variables that are defined in the Window Editor.

Use $[\text{WINDOW}]$ to display the Window Editor. You can either:

- Enter appropriate values.
  — or —
- Select 9:ZoomData from the $[\text{F2}]$ Zoom toolbar menu. (Although you can use any zoom, ZoomData is optimized for st plots.)

**ZoomData** sets the viewing window to display all statistical data points.

For histograms and box plots, only xmin and xmax are adjusted. If the top of a histogram is not shown, trace the histogram to find the value for ymax.

**Note:** $[\text{F2}]$ Zoom is available on the Y= Editor, Window Editor, and Graph screen.
Changing the Graph Format

Press:

[F1] 9

— or —

from the Y= Editor, Window Editor, or Graph screen.

Then change the settings as necessary.

Tracing a Stat Plot

From the Graph screen, press [F3] to trace a plot. The movement of the trace cursor depends on the Plot Type.

<table>
<thead>
<tr>
<th>Plot Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scatter or xyline</td>
<td>Tracing begins at the first data point.</td>
</tr>
<tr>
<td>Box plot</td>
<td>Tracing begins at the median. Press ① to trace to Q1 and minX. Press ② to trace to Q3 and maxX.</td>
</tr>
<tr>
<td>Histogram</td>
<td>The cursor moves from the top center of each bar, starting from the leftmost bar.</td>
</tr>
</tbody>
</table>

Note: When a stat plot is displayed, the Graph screen does not automatically pan if you trace off the left or right side of the screen. However, you can still press [ENTER] to center the screen on the trace cursor.
When you press ↺ or ↺ to move to another plot or y(x) function, tracing moves to the current or beginning point on that plot (not to the nearest pixel).

**Using Frequencies and Categories**

To manipulate the way in which data points are analyzed, you can use frequency values and/or category values. Frequency values let you “weight” particular data points. Category values let you analyze a subset of the data points.

**Example of a Frequency Column**

In a data variable, you can use any column in the Data/Matrix Editor to specify a frequency value (or weight) for the data points on each row. A frequency value must be an integer ≥ 0 if Calculation Type = OneVar or MedMed or if Plot Type = Box Plot. For other statistical calculations or plots, the frequency value can be any number ≥ 0.

For example, suppose you enter a student’s test scores, where:

- The mid-semester exam is weighted twice as much as other tests.
- The final exam is weighted three times as much.
In the Data/Matrix Editor, you can enter the test scores and frequency values in two columns.

<table>
<thead>
<tr>
<th>Test scores</th>
<th>Frequency values</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>c2</td>
</tr>
<tr>
<td>85</td>
<td>1</td>
</tr>
<tr>
<td>97</td>
<td>1</td>
</tr>
<tr>
<td>92</td>
<td>2</td>
</tr>
<tr>
<td>89</td>
<td>1</td>
</tr>
<tr>
<td>91</td>
<td>1</td>
</tr>
<tr>
<td>95</td>
<td>3</td>
</tr>
</tbody>
</table>

These weighted scores are equivalent to the single column of scores listed to the right.

<table>
<thead>
<tr>
<th>c1</th>
</tr>
</thead>
<tbody>
<tr>
<td>85</td>
</tr>
<tr>
<td>97</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>92</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>92</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>89</th>
</tr>
</thead>
<tbody>
<tr>
<td>91</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>95</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>95</td>
<td>2</td>
</tr>
</tbody>
</table>

| 95 | 2 |

Note: A frequency value of 0 effectively removes the data point from analysis.

To use frequency values, specify the frequency column when you perform a statistical calculation or define a stat plot. For example:
Note: You can also use frequency values from a list variable instead of a column.

**Example of a Category Column**

In a data variable, you can use any column to specify a category (or subset) value for the data points on each row. A category value can be any number.

Suppose you enter the test scores from a class that has 10th and 11th grade students. You want to analyze the scores for the whole class, but you also want to analyze categories such as 10th grade girls, 10th grade boys, 10th grade girls and boys, etc.

First, determine the category values you want to use.

<table>
<thead>
<tr>
<th>Category Value</th>
<th>Used to indicate:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10th grade girl</td>
</tr>
<tr>
<td>2</td>
<td>10th grade boy</td>
</tr>
<tr>
<td>3</td>
<td>11th grade girl</td>
</tr>
<tr>
<td>4</td>
<td>11th grade boy</td>
</tr>
</tbody>
</table>

**Note:** You do not need a category value for the whole class. Also, you do not need category values for all 10th graders or all 11th graders since they are combinations of other categories.
In the Data/Matrix Editor, you can enter the scores and the category values in two columns.

<table>
<thead>
<tr>
<th>Test scores</th>
<th>Category values</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>c2</td>
</tr>
<tr>
<td>85</td>
<td>1</td>
</tr>
<tr>
<td>97</td>
<td>3</td>
</tr>
<tr>
<td>92</td>
<td>2</td>
</tr>
<tr>
<td>88</td>
<td>3</td>
</tr>
<tr>
<td>90</td>
<td>2</td>
</tr>
<tr>
<td>95</td>
<td>1</td>
</tr>
<tr>
<td>79</td>
<td>4</td>
</tr>
<tr>
<td>68</td>
<td>2</td>
</tr>
<tr>
<td>92</td>
<td>4</td>
</tr>
<tr>
<td>84</td>
<td>3</td>
</tr>
<tr>
<td>82</td>
<td>1</td>
</tr>
</tbody>
</table>

To use category values, specify the category column and the category values to include in the analysis when you perform a statistical calculation or define a stat plot.
Note: You can also use category values from a list variable instead of a column.

<table>
<thead>
<tr>
<th>To analyze:</th>
<th>Include Categories:</th>
</tr>
</thead>
<tbody>
<tr>
<td>10th grade girls</td>
<td>{1}</td>
</tr>
<tr>
<td>10th grade boys</td>
<td>{2}</td>
</tr>
<tr>
<td>10th grade girls and boys</td>
<td>{1,2}</td>
</tr>
<tr>
<td>11th grade girls</td>
<td>{3}</td>
</tr>
<tr>
<td>11th grade boys</td>
<td>{4}</td>
</tr>
<tr>
<td>11th grade girls and boys</td>
<td>{3,4}</td>
</tr>
<tr>
<td>all girls (10th and 11th)</td>
<td>{1,3}</td>
</tr>
<tr>
<td>all boys (10th and 11th)</td>
<td>{2,4}</td>
</tr>
</tbody>
</table>

Note: To analyze the whole class, leave the Category input box blank. Any category values are ignored.
The Calculator-Based Laboratory™ System (CBL 2) and Calculator-Based Ranger™ System (CBR) are optional accessories, available separately, that let you collect data from a variety of real-world experiments. TI-89 Titanium, CBL 2 and CBR programs are available from the TI web site at education.ti.com.

How CBL 2™ Data Is Stored

When you collect data with the CBL 2, that data is initially stored in the CBL 2 unit itself. You must then retrieve the data (transfer it to the TI-89 Titanium) by using the Get command, which is described in the Technical Reference module.

Although each set of retrieved data can be stored in several variable types (list, real, matrix, pic), using list variables makes it easier to perform statistical calculations.

When you transfer the collected information to the TI-89 Titanium, you can specify the list variable names that you want to use.

For example, you can use the CBL 2 to collect temperature data over a period of time. When you transfer the data, suppose you store:

- Temperature data in a list variable called temp.
- Time data in a list variable called time.

After you store the CBL 2 information on the TI-89 Titanium, there are two ways to use the CBL 2 list variables.
Note: For specifics about using the CBL 2 and retrieving data to the TI-89 Titanium, refer to the guidebook that comes with the CBL 2 unit.

**Referring to the CBL 2™ Lists**

When you perform a statistical calculation or define a plot, you can refer explicitly to the CBL 2 list variables. For example:

Type the CBL 2 list variable name instead of a column number.

**Creating a Data Variable with the CBL 2™ Lists**

You can create a new data variable that consists of the necessary CBL 2 list variables.
• From the Home screen or a program, use the **NewData** command.

```plaintext
NewData dataVar, list1 [,list2 ] [,list3 ] ...
```

—CBL 2 list variable names. In the new data variable, list1 will be copied to column 1, list 2 to column 2, etc.

—Name of the new data variable that you want to create.

For example:

```plaintext
NewData temp1, time, temp
```

creates a data variable called `temp1` in which time is in column 1 and `temp` is in column 2.

• From the Data/Matrix Editor, create a new, empty data variable with the applicable name. For each CBL 2 list that you want to include, define a column header as that list name.

For example, define column 1 as time, column 2 as temp.

![Data/Matrix Editor](image)

**Note:** To define or clear a column header, use `F4`. For more information, refer to the *Data/Matrix Editor* module.

At this point, the columns are linked to the CBL 2 lists. If the lists are changed, the columns will be updated automatically. However, if the lists are deleted, the data will be lost.

To make the data variable independent of the CBL 2 lists, clear the column header for each column. The information remains in the column, but the column is no longer linked to the CBL 2 list.
You can also use the Calculator-Based Ranger™ (CBR) to explore the mathematical and scientific relationships between distance, velocity, acceleration, and time using data collected from activities you perform.
Programming

Running an Existing Program

After a program is created (as described in the remaining sections of this module), you can run it from the Home screen. The program’s output, if any, is displayed on the Program I/O screen, in a dialog box, or on the Graph screen.

Running a Program

On the Home screen:

1. Type the name of the program.

2. You must always type a set of parentheses after the name. Some programs require you to pass an argument to the program.

   **Note:** Use [2nd] [VAR-LINK] to list existing PRGM variables. Highlight a variable and press [ENTER] to paste its name to the entry line.

3. Press [ENTER].
Note: Arguments specify initial values for a program.

When you run a program, the TI-89 Titanium automatically checks for errors. For example, the following message is displayed if you:

- Do not enter ( ) after the program name.

This error message appears if you:

- Do not enter enough arguments, if required.

To cancel program execution if an error occurs, press [ESC]. You can then correct any problems and run the program again.

Note: The TI-89 Titanium also checks for run-time errors that are found within the program itself.

“Breaking” a Program

When a program is running, the BUSY indicator is displayed in the status line.
Press \textbf{ON} to stop program execution. A message is then displayed.

- To display the program in the Program Editor, press \textbf{[ENTER]}. The cursor appears at the command where the break occurred.
- To cancel program execution, press \textbf{ESC}.

**Where Is the Output Displayed?**

Depending on the commands in the program, the TI-89 Titanium automatically displays information on the applicable screen.

- Most output and input commands use the Program I/O screen. (Input commands prompt the user to enter information.)
- Graph-related commands typically use the Graph screen.

After the program stops, the TI-89 Titanium shows the last screen that was displayed.

**The Program I/O Screen**

On the Program I/O screen, new output is displayed below any previous output (which may have been displayed earlier in the same program or a different program). After a full page of output, the previous output scrolls off the top of the screen.
Note: To clear any previous output, enter the ClrIO command in your program. You can also execute ClrIO from the Home screen.

When a program stops on the Program I/O screen, you need to recognize that it is not the Home screen (although the two screens are similar). The Program I/O screen is used only to display output or to prompt the user for input. You cannot perform calculations on this screen.

Note: If Home screen calculations don’t work after you run a program, you may be on the Program I/O screen.

Leaving the Program I/O Screen

From the Program I/O screen:

• Press F5 to toggle between the Home screen and the Program I/O screen.
  – or –

• Press ESC, 2nd [QUIT], or HOME to display the Home screen.
  – or –

• Display any other application screen (with APPS, [Y=], etc.).
Starting a Program Editor Session

Each time you start the Program Editor, you can resume the current program or function (that was displayed the last time you used the Program Editor), open an existing program or function, or start a new program or function.

Starting a New Program or Function

1. Press \[\text{APPs}\] and then select \text{Program Editor}.

2. Select 3:\text{New}.

3. Specify the applicable information for the new program or function.

<table>
<thead>
<tr>
<th>Item</th>
<th>Lets you:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Select whether to create a new program or function.</td>
</tr>
<tr>
<td>Folder</td>
<td>Select the folder in which the new program or function will be stored. For information about folders, refer to the \textit{Calculator Home Screen} module.</td>
</tr>
</tbody>
</table>
### Variable
Type a variable name for the program or function. If you specify a variable that already exists, an error message will be displayed when you press [ENTER]. When you press [ESC] or [ENTER] to acknowledge the error, the **NEW** dialog box is redisplayed.

4. Press [ENTER] (after typing in an input box such as **Variable**, you must press [ENTER] twice) to display an empty “template.”

This is the template for a program. Functions have a similar template.

You can now use the Program Editor as described in the remaining sections of this module.

**Note:** A program (or function) is saved automatically as you type. You do not need to save it manually before leaving the Program Editor, starting a new program, or opening a previous one.

---

### Resuming the Current Program

You can leave the Program Editor and go to another application at any time. To return to the program or function that was displayed when you left the Program Editor, launch Program Editor again and select **1:Current**.
Starting a New Program from the Program Editor

To leave the current program or function and start a new one:

1. Press $\text{F1}$ and select $\text{3:New}$.
2. Specify the type, folder, and variable for the new program or function.
3. Press $\text{ENTER}$ twice.

Opening a Previous Program

You can open a previously created program or function at any time.

1. From within the Program Editor, press $\text{F1}$ and select $\text{1:Open}$.
   – or –
   From another application, launch Program Editor again and select $\text{2:Open}$.

2. Select the applicable type, folder, and variable.
3. Press $\text{ENTER}$.

Note: By default, Variable shows the first existing program or function in alphabetical order.
**Copying a Program**

In some cases, you may want to copy a program or function so that you can edit the copy while retaining the original.

1. Display the program or function you want to copy.
2. Press 2nd and select **2:Save Copy As**.
3. Specify the folder and variable for the copy.
4. Press **ENTER** twice.

**Note about Deleting a Program**

Because all Program Editor sessions are saved automatically, you can accumulate quite a few previous programs and functions, which take up memory storage space.

To delete programs and functions, use the VAR-LINK screen (2nd [VAR-LINK]). For information about VAR-LINK, refer to the *Memory and Variable Management* module.

**Overview of Entering a Program**

A program is a series of commands executed in sequential order (although some commands alter the program flow). In general, anything that can be executed from the Home screen can be included in a program. Program execution continues until it reaches the end of the program or a **Stop** command.
**Entering and Editing Program Lines**

On a blank template, you can begin entering commands for your new program.

Program name, which you specify when you create a new program.

Enter your program commands between **Prgm** and **EndPrgm**.

All program lines begin with a colon.

**Note**: Use the cursor pad to scroll through the program for entering or editing commands. Use ↑↓ or ←→ to go to the top or bottom of a program, respectively.

You enter and edit program commands in the Program Editor by using the same techniques used to enter and edit text in the Text Editor. Refer to “Entering and Editing Text” in the *Text Editor* module.

After typing each program line, press **ENTER**. This inserts a new blank line and lets you continue entering another line. A program line can be longer than one line on the screen; if so, it will wrap to the next screen line automatically.

**Note**: Entering a command does not execute that command. It is not executed until you run the program.

**Entering Multi-Command Lines**

To enter more than one command on the same line, separate them with a colon by pressing **2nd [:]**.
**Entering Comments**

A comment symbol (\(\text{©}\)) lets you enter a remark in a program. When you run the program, all characters to the right of \(\text{©}\) are ignored.

---

:progl()
:Prgm
\(\text{©}\):Displays sum of 1 thru n
:Request "Enter an integer",n
\(\text{©}\):expr(n)\(\rightarrow\)n:Convert to numeric expression
:------

---

1. Description of the program.
2. Description of `expr`.

**Note:** Use comments to enter information that is useful to someone reading the program code.

To enter the comment symbol, press:

- \(\text{©}\)
- \(\text{©}\)
  - or -
- Press \(\text{F2}\) and select \(\text{©}\)

**Controlling the Flow of a Program**

When you run a program, the program lines are executed in sequential order. However, some commands alter the program flow. For example:
• Control structures such as `If...EndIf` commands use a conditional test to decide which part of a program to execute.

• Loops commands such as `For...EndFor` repeat a group of commands.

**Using Indentation**

For more complex programs that use `If...EndIf` and loop structures such as `For...EndFor`, you can make the programs easier to read and understand by using indentation.

```
:If x>5 Then
  : Disp "x is > 5"
:Else
  : Disp "x is < or = 5"
:EndIf
```

**Displaying Calculated Results**

In a program, calculated results are not displayed unless you use an output command. This is an important difference between performing a calculation on the Home screen and in a program.

These calculations will not display a result in a program (although they will on the Home screen).

```
:12*6
:cos(π/4)
:solve(x^2-x-2=0,x)
```

Output commands such as `Disp` will display a result in a program.

```
:Disp 12*6
:Disp cos(π/4)
:Disp solve(x^2-x-2=0,x)
```
Displaying a calculation result does not store that result. If you need to refer to a result later, store it to a variable.

Note: A list of output commands is available.

Getting Values into a Program

To input values into a program, you can:

• Require the users to store a value (with \texttt{STO\textup{\texttt{)}} to the necessary variables before running the program. The program can then refer to these variables.

• Enter the values directly into the program itself.

• Include input commands that prompt the users to enter the necessary values when they run the program.

• Require the users to pass one or more values to the program when they run it.

Note: A list of input commands is available.
Example of Passing Values to a Program

The following program draws a circle on the Graph screen and then draws a horizontal line across the top of the circle. Three values must be passed to the program: x and y coordinates for the circle’s center and the radius r.

• When you write the program in the Program Editor:

In the ( ) beside the program name, specify the variables that will be used to store the passed values.

Notice that the program also contains commands that set up the Graph screen.

:circ(x,y,r) ❶
Prgm
:FnOff
:ZoomStd
:ZoomSqr
:Circle x,y,r
:LineHorz y+r
:EndPrgm

❶ Only circ( ) is initially displayed on the blank template; be sure to edit this line.

Note: In this example, you cannot use circle as the program name because it conflicts with a command name.

Before drawing the circle, the program turns off any selected Y= Editor functions, displays a standard viewing window, and “squares” the window.
To run the program from the Home screen:

The user must specify the applicable values as arguments within the ( ).

\[
\text{circ}(0,0,5)
\]

Passed to r
Passed to y
Passed to x

The arguments, in order, are passed to the program.

Note: This example assumes that the user enters values that can be displayed by the viewing window set up by \texttt{ZoomStd} and \texttt{ZoomSqr}.

**Overview of Entering a Function**

A function created in the Program Editor is very similar to the functions and instructions that you typically use from the Home screen.

**Why Create a User-Defined Function?**

Functions (as well as programs) are ideal for repetitive calculations or tasks. You only need to write the function once. Then you can reuse it as many times as necessary. Functions, however, have some advantages over programs.
• You can create functions that expand on the TI-89 Titanium’s built-in functions. You can then use the new functions the same as any other function.

• Functions return values that can be graphed or entered in a table; programs cannot.

• You can use a function (but not a program) within an expression. For example: \( 3*\text{func1}(3) \) is valid, but not \( 3*\text{prog1}(3) \).

• Because you pass arguments to a function, you can write generic functions that are not tied to specific variable names.

**Note:** You can create a function from the Home screen, but the Program Editor is more convenient for complex, multi-line functions.

**Differences Between Functions and Programs**

This guidebook sometimes uses the word command as a generic reference to instructions and functions. When writing a function, however, you must differentiate between instructions and functions.

A user-defined function:

• Can use the following instructions only. Any others are invalid.

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Define</th>
<th>Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>For...EndFor</td>
<td>Goto</td>
<td>If...EndIf (all forms)</td>
</tr>
<tr>
<td>Lbl</td>
<td>Local</td>
<td>Loop...EndLoop</td>
</tr>
<tr>
<td>Return</td>
<td>While...EndWhile</td>
<td>( \text{\textlangle STO\rangle} ) key)</td>
</tr>
</tbody>
</table>
• Can use all built-in TI-89 Titanium / Voyage™ 200 functions except:

<table>
<thead>
<tr>
<th>setFold</th>
<th>setGraph</th>
<th>setMode</th>
</tr>
</thead>
<tbody>
<tr>
<td>setTable</td>
<td>switch</td>
<td></td>
</tr>
</tbody>
</table>

• Can refer to any variable; however, it can store a value to a local variable only.
  - The arguments used to pass values to a function are treated as local variables automatically. If you store to any other variables, you must declare them as local from within the function.

• Cannot call a program as a subroutine, but it can call another user-defined function.
• Cannot define a program.
• Cannot define a global function, but it can define a local function.

**Note:** Information about local variables is available.

**Entering a Function**

When you create a new function in the Program Editor, the TI-89 Titanium displays a blank “template.”

**Note:** Use the cursor pad to scroll through the function for entering or editing commands.
If the function requires input, one or more values must be passed to the function. (A user-defined function can store to local variables only, and it cannot use instructions that prompt the user for input.)

**How to Return a Value from a Function**

There are two ways to return a value from a function:

- As the last line in the function (before `EndFunc`), calculate the value to be returned.

```plaintext
: cube (x)
: Func
: x^3
: EndFunc
```
• Use **Return**. This is useful for exiting a function and returning a value at some point other than the end of the function.

```
cube(x)
:Func
:If x<0
 : Return 0
 : x^3
:EndFunc
```

**Note:** This example calculates the cube if $x \geq 0$; otherwise, it returns 0.

The argument $x$ is automatically treated as a local variable. However, if the example needed another variable, the function would need to declare it as local by using the **Local** command.

There is an implied **Return** at the end of the function. If the last line is not an expression, an error occurs.

**Example of a Function**

The following function returns the $x$th root of a value $y$ ($x\sqrt{y}$). Two values must be passed to the function: $x$ and $y$. 
Note: Because x and y in the function are local, they are not affected by any existing x or y variable.

<table>
<thead>
<tr>
<th>Function as called from the Home Screen</th>
<th>Function as defined in the Program Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>3⇒x:125⇒y</td>
<td>:xroot(x,y)</td>
</tr>
<tr>
<td>4*xroot(3,125)</td>
<td>:Func</td>
</tr>
<tr>
<td></td>
<td>:y^(1/x)</td>
</tr>
<tr>
<td></td>
<td>:EndFunc</td>
</tr>
</tbody>
</table>

### Calling One Program from Another

One program can call another program as a subroutine. The subroutine can be external (a separate program) or internal (included in the main program). Subroutines are useful when a program needs to repeat the same group of commands at several different places.
Calling a Separate Program

To call a separate program, use the same syntax used to run the program from the Home screen.

```plaintext
subtest1()
: Prgm
: For i, 1, 4, 1
:  subtest2(i, i*1000)
: EndFor
: EndPrgm
```

Calling an Internal Subroutine

To define an internal subroutine, use the `Define` command with `Prgm...EndPrgm`. Because a subroutine must be defined before it can be called, it is a good practice to define subroutines at the beginning of the main program.

```plaintext
subtest2(x, y)
: Prgm
: Disp x, y
: EndPrgm
```
An internal subroutine is called and executed in the same way as a separate program.

```
:subtest1()
:Prgm
① :local subtest2
② :Define subtest2(x,y)=Prgm
: : Disp x,y
② :EndPrgm
"Beginning of main program
:For i,1,4,1
③ : subtest2(i,I*1000)
:EndFor
:EndPrgm
```

① Declares the subroutine as a local variable.
② Defines the subroutine.
③ Calls the subroutine.

**Note:** Use the Program Editor’s F4 Var toolbar menu to enter the Define and Prgm...EndPrgm commands.

**Notes about Using Subroutines**

At the end of a subroutine, execution returns to the calling program. To exit a subroutine at any other time, use the **Return** command.

A subroutine cannot access local variables declared in the calling program. Likewise, the calling program cannot access local variables declared in a subroutine.
**Lbl** commands are local to the programs in which they are located. Therefore, a **Goto** command in the calling program cannot branch to a label in a subroutine or vice versa.

**Using Variables in a Program**

Programs use variables in the same general way that you use them from the Home screen. However, the “scope” of the variables affects how they are stored and accessed.

**Scope of Variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System (Global) Variables</td>
<td>Variables with reserved names that are created automatically to store data about the state of the calculator. For example, Window variables (<em>xmin</em>, <em>xmax</em>, <em>ymin</em>, <em>ymax</em>, etc.) are globally available from any folder.</td>
</tr>
<tr>
<td></td>
<td>• You can always refer to these variables by using the variable name only, regardless of the current folder.</td>
</tr>
<tr>
<td></td>
<td>• A program cannot create system variables, but it can use the values and (in most cases) store new values.</td>
</tr>
</tbody>
</table>
### Folder Variables
Variables that are stored in a particular folder.

- If you store to a variable name only, it is stored in the current folder. For example:
  \[ 5 \rightarrow start \]
- If you refer to a variable name only, that variable must be in the current folder. Otherwise, it cannot be found (even if the variable exists in a different folder).
- To store or refer to a variable in another folder, you must specify a path name. For example:

\[
5 \rightarrow \text{class}\backslash start \\
(\text{class} = \text{Variable name}; \text{start} = \text{Folder Name})
\]

After the program stops, any folder variables created by the program still exist and still take up memory.

### Local Variables
Temporary variables that exist only while a program is running. When the program stops, local variables are deleted automatically.

- To create a local variable in a program, use the **Local** command to declare the variable.
- A local variable is treated as unique even if there is an existing folder variable with the same name.
- Local variables are ideal for temporarily storing values that you do not want to save.
Note: If a program has local variables, a graphed function cannot access them. For example:

Local a
5⇒a
Graph a*cos(x)

may display an error or an unexpected result (if a is an existing variable in the current folder).

Circular Definition Errors

When evaluating a user-defined function or running a program, you can specify an argument that includes the same variable that was used to define the function or create the program. However, to avoid Circular definition errors, you must assign a value for x or i variables that are used in evaluating the function or running the program. For example:

1 x+1⇒x

– or –

For i,i,10,1
1 Disp i
1 EndFor

Causes a Circular definition error message if x or i does not have a value. The error does not occur if x or i has already been assigned a value.
### Variable-Related Commands and Functions

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STO►</strong> key</td>
<td>Stores a value to a variable. As on the Home screen, pressing <strong>STO►</strong> enters a ▶ symbol.</td>
</tr>
<tr>
<td>Archive</td>
<td>Moves specified variables from RAM to user data archive memory.</td>
</tr>
<tr>
<td>BldData</td>
<td>Lets you create a data variable based on the graph information entered in the Y= Editor, Window Editor, etc.</td>
</tr>
<tr>
<td>CopyVar</td>
<td>Copies the contents of a variable.</td>
</tr>
<tr>
<td>Define</td>
<td>Defines a program (subroutine) or function variable within a program.</td>
</tr>
<tr>
<td>DelFold</td>
<td>Deletes a folder. All variables in that folder must be deleted first.</td>
</tr>
<tr>
<td>DelType</td>
<td>Deletes unarchived variables of the specified type in all folders.</td>
</tr>
<tr>
<td>DelVar</td>
<td>Deletes a variable.</td>
</tr>
<tr>
<td>getFold</td>
<td>Returns the name of the current folder.</td>
</tr>
<tr>
<td>getType</td>
<td>Returns a string that indicates the data type (EXPR, LIST, etc.) of a variable.</td>
</tr>
<tr>
<td>isArchiv()</td>
<td>Indicates if the variable is archived or not.</td>
</tr>
<tr>
<td>isLocked()</td>
<td>Indicates if the variable is locked or not.</td>
</tr>
<tr>
<td>isVar()</td>
<td>Indicates if the variable is in the symbol table or not.</td>
</tr>
<tr>
<td>Local</td>
<td>Declares one or more variables as local variables.</td>
</tr>
</tbody>
</table>
### Using Local Variables in Functions or Programs

A local variable is a temporary variable that exists only while a user-defined function is being evaluated or a user-defined program is running.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td>Locks a variable so that it cannot be accidentally changed or deleted without first being unlocked.</td>
</tr>
<tr>
<td>MoveVar</td>
<td>Moves a variable from one folder to another.</td>
</tr>
<tr>
<td>NewData</td>
<td>Creates a data variable whose columns consist of a series of specified lists.</td>
</tr>
<tr>
<td>NewFold</td>
<td>Creates a new folder.</td>
</tr>
<tr>
<td>NewPic</td>
<td>Creates a picture variable based on a matrix.</td>
</tr>
<tr>
<td>Rename</td>
<td>Renames a variable.</td>
</tr>
<tr>
<td>Unarchiv</td>
<td>Moves specified variables from user data archive memory to RAM.</td>
</tr>
<tr>
<td>Unlock</td>
<td>Unlocks a locked variable.</td>
</tr>
</tbody>
</table>

Note: The Define, DelVar, and Local commands are available from the Program Editor’s [F4] Var toolbar menu.
Example of a Local Variable

The following program segment shows a **For...EndFor loop** (which is discussed later in this module). The variable $i$ is the loop counter. In most cases, the variable $i$ is used only while the program is running.

```
1: Local I
   : For i,0,5,1
     : Disp I
   : EndFor
   : Disp i
```

1 Declares variable $i$ as local.

**Note:** As often as possible, use local variables for any variable that is used only within a program and does not need to be stored after the program stops.

If you declare variable $i$ as local, it is deleted automatically when the program stops so that it does not use up memory.

**What Causes an Undefined Variable Error Message?**

An **Undefined** variable error message displays when you evaluate a user-defined function or run a user-defined program that references a local variable that is not initialized (assigned a value).

This example is a multi-statement function, rather than a program. Line breaks are shown here, but you would type the text in the entry line as one continuous line, such as: `Define fact(n)=Func:Local...` where the ellipsis indicates the entry line text continues off-screen.
For example:

```
Define fact(n)=Func:
  Local m:
  While n>1:
    n*m->m: n-1->n:
  EndWhile:
  Return m:
EndFunc
```

Local variable \( m \) is not assigned an initial value.

In the example above, the local variable \( m \) exists independently of any variable \( m \) that exists outside of the function.

**You Must Initialize Local Variables**

All local variables must be assigned an initial value before they are referenced.

```
Define fact(n)=Func:
  Local m: 1->m:
  While n>1:
    n*m->m: n-1->n:
  EndWhile:
  Return m:
EndFunc
```

1 is stored as the initial value for \( m \).

The calculator cannot use a local variable to perform symbolic calculations.
To Perform Symbolic Calculations

If you want a function or program to perform symbolic calculations, you must use a global variable instead of a local. However, you must be certain that the global variable does not already exist outside of the program. The following methods can help.

• Refer to a global variable name, typically with two or more characters, that is not likely to exist outside of the function or program.

• Include DelVar within the function or program to delete the global variable, if it exists, before referring to it. (DelVar does not delete locked or archived variables.)

String Operations

Strings are used to enter and display text characters. You can type a string directly, or you can store a string to a variable.
How Strings Are Used

A string is a sequence of characters enclosed in "quotes." In programming, strings allow the program to display information or prompt the user to perform some action. For example:

```
Disp "The result is",answer

– or –

Input "Enter the angle in degrees",ang1

– or –

"Enter the angle in degrees"→str1
Input str1,ang1
```

Some input commands (such as InputStr) automatically store user input as a string and do not require the user to enter quotation marks.

A string cannot be evaluated mathematically, even if it appears to be a numeric expression. For example, the string "61" represents the characters "6" and "1", not the number 61.

Although you cannot use a string such as "61" or "2x+4" in a calculation, you can convert a string into a numeric expression by using the expr command.
# String Commands

**Note:** See the *Technical Reference* module for syntax for all commands and functions.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Converts a string into a variable name. This is called indirection.</td>
</tr>
<tr>
<td>&amp;</td>
<td>Appends (concatenates) two strings into one string.</td>
</tr>
<tr>
<td>char</td>
<td>Returns the character that corresponds to a specified character code. This is the opposite of the <code>ord</code> command.</td>
</tr>
<tr>
<td>dim</td>
<td>Returns the number of characters in a string.</td>
</tr>
<tr>
<td>expr</td>
<td>Converts a string into an expression and executes that expression. This is the opposite of the <code>string</code> command. <strong>Important:</strong> Some user input commands store the entered value as a string. Before you can perform a mathematical operation on that value, you must convert it to a numeric expression.</td>
</tr>
<tr>
<td>format</td>
<td>Returns an expression as a character string based on the format template (fixed, scientific, engineering, etc.)</td>
</tr>
<tr>
<td>inString</td>
<td>Searches a string to see if it contains a specified substring. If so, <code>inString</code> returns the character position where the first occurrence of the substring begins.</td>
</tr>
<tr>
<td>left</td>
<td>Returns a specified number of characters from the left side (beginning) of a string.</td>
</tr>
<tr>
<td>mid</td>
<td>Returns a specified number of characters from any position within a string.</td>
</tr>
<tr>
<td>ord</td>
<td>Returns the character code of the first character within a string. This is the opposite of the <code>char</code> command.</td>
</tr>
</tbody>
</table>
Conditional Tests

Conditional tests let programs make decisions. For example, depending on whether a test is true or false, a program can decide which of two actions to perform. Conditional tests are used with control structures such as \texttt{If...EndIf} and loops such as \texttt{While...EndWhile} (described later in this module).

\begin{tabular}{|l|p{0.9\textwidth}|}
\hline
\textbf{Command} & \textbf{Description} \\
\hline
\texttt{right} & Returns a specified number of characters from the right side (end) of a string. \\
\hline
\texttt{rotate} & Rotates the characters in a string. The default is \texttt{-1} (rotate right one character). \\
\hline
\texttt{shift} & Shifts the characters in a string and replaces them with spaces. The default is \texttt{-1} (shift right one character and replace with one space). Examples: \texttt{shift("abcde",2)}⇒"cde " and \texttt{shift("abcde")}⇒" abcd" \\
\hline
\texttt{string} & Converts a numeric expression into a string. This is the opposite of the \texttt{expr} command. \\
\hline
\end{tabular}
Entering a Test Operator

- Type the operator directly from the keyboard.
  – or –

- Press \texttt{2nd} [MATH] and select \texttt{8:Test}. Then select the operator from the menu.
  – or –

- Display the built-in functions. Press: \texttt{CATALOG}\texttt{The test operators are listed near the bottom of the [F2] Built-in menu.}

Relational Tests

Relational operators let you define a conditional test that compares two values. The values can be numbers, expressions, lists, or matrices (but they must match in type and dimension).

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if:</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>Greater than</td>
<td>a&gt;8</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
<td>a&lt;0</td>
</tr>
<tr>
<td>≥</td>
<td>Greater than or equal to</td>
<td>a+b≥100</td>
</tr>
<tr>
<td>≤</td>
<td>Less than or equal to</td>
<td>a+6≤b+1</td>
</tr>
<tr>
<td>=</td>
<td>Equal</td>
<td>list1=list2</td>
</tr>
<tr>
<td>≠</td>
<td>Not equal to</td>
<td>mat1≠mat2</td>
</tr>
</tbody>
</table>
**Note:** From the keyboard, you can type:

>= for ≥
<= for ≤
/= for ≠
(To get the / character, press ÷.)

**Boolean Tests**

Boolean operators let you combine the results of two separate tests.

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if:</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>Both tests are true</td>
<td>a&gt;0 and a≤10</td>
</tr>
<tr>
<td>or</td>
<td>At least one test is true</td>
<td>a≤0 or b+c&gt;10</td>
</tr>
<tr>
<td>xor</td>
<td>One test is true and the other is false</td>
<td>a+6&lt;b+1 xor c&lt;d</td>
</tr>
</tbody>
</table>

**The Not Function**

The **not** function changes the result of a test from true to false and vice versa. For example:

not x>2   is true if   x≤2
           is false if  x>2

**Note:** If you use **not** from the Home screen, it is shown as ~ in the history area. For example, not x>2 is shown as ~(x>2).
An If...EndIf structure uses a conditional test to decide whether or not to execute one or more commands. Lbl (label) and Goto commands can also be used to branch (or jump) from one place to another in a program.

**F2 Control Toolbar Menu**

To enter If...EndIf structures, use the Program Editor’s [F2] Control toolbar menu.

The If command is available directly from the [F2] menu.

To see a submenu that lists other If structures, select 2:If...Then.

When you select a structure such as If...Then...EndIf, a template is inserted at the cursor location.

The cursor is positioned so that you can enter a conditional test.
If Command

To execute only one command if a conditional test is true, use the general form:

```plaintext
:If x>5
①: Disp "x is greater than 5"
②:Disp x
```

① Executed only if x>5; otherwise, skipped.
② Always displays the value of x.

In this example, you must store a value to x before executing the If command.

**Note:** Use indentation to make your programs easier to read and understand.

If...Then...EndIf Structures

To execute one group of commands if a conditional test is true, use the structure:

```plaintext
:If x>5 Then
①: Disp "x is greater than 5"
①: 2*x→x
②:EndIf
 :Disp x
```

① Executed only if x>5.
② Displays value of:
  • 2x if x>5
  • x if x≤5
Note: EndIf marks the end of the Then block that is executed if the condition is true.

### If...Then...Else... EndIf Structures

To execute one group of commands if a conditional test is true and a different group if the condition is false, use this structure:

```
:If x>5 Then
  1:  Disp "x is greater than 5"
  1:  2*x→x
Else
  2:  Disp "x is less than or equal to 5"
    :  5*x→x
:EndIf
:Disp x
```

1. Executed only if $x > 5$.
2. Executed only if $x \leq 5$.
3. Displays value of:
   - $2x$ if $x > 5$
   - $5x$ if $x \leq 5$

### If...Then...ElseIf... EndIf Structures

A more complex form of the If command lets you test a series of conditions. Suppose your program prompts the user for a number that corresponds to one of four options. To test for each option (If Choice=1, If Choice = 2, etc.), use the If...Then...ElseIf...EndIf structure.
Refer to the *Technical Reference* module for more information and an example.

**Lbl and Goto Commands**

You can also control the flow of your program by using **Lbl** (label) and **Goto** commands.

Use the **Lbl** command to label (assign a name to) a particular location in the program.

```
Lbl  labelName
    ______ name to assign to this location (use the same naming convention as a variable name)
```

You can then use the **Goto** command at any point in the program to branch to the location that corresponds to the specified label.

```
Goto labelName
    ______ specifies which **Lbl** command to branch to
```
Because a **Goto** command is unconditional (it always branches to the specified label), it is often used with an **If** command so that you can specify a conditional test. For example:

```
:If x>5
  :  Goto GT5
  :  Disp x
  :--------
  :--------
  :Lbl GT5
  :Disp "The number was > 5"
```

1. If $x > 5$, branches directly to label GT5.
2. For this example, the program must include commands (such as **Stop**) that prevent **Lbl** GT5 from being executed if $x \leq 5$.

**Using Loops to Repeat a Group of Commands**

To repeat the same group of commands successively, use a loop. Several types of loops are available. Each type gives you a different way to exit the loop, based on a conditional test.

**F2 Control Toolbar Menu**

To enter most of the loop-related commands, use the Program Editor’s [F2] **Control** toolbar menu.
When you select a loop, the loop command and its corresponding End command are inserted at the cursor location.

You can then begin entering the commands that will be executed in the loop.

**Note:** A loop command marks the start of the loop. The corresponding End command marks the end of the loop.

**For...EndFor Loops**

A For...EndFor loop uses a counter to control the number of times the loop is repeated. The syntax of the For command is:

**Note:** The ending value can be less than the beginning value, but the increment must be negative.

\[
\text{For}(\text{variable, begin, end [ , increment]}))
\]

1. variable used as a counter
2. counter value used the first time For is executed
3. exits the loop when variable exceeds this value
4. added to the counter each subsequent time For is executed
When `For` is executed, the variable value is compared to the end value. If variable does not exceed end, the loop is executed; otherwise, program control jumps to the command following `EndFor`.

\[ i > 5 \quad i \leq 5 \quad : \text{For } i,0,5,1 \quad : \text{Disp } I \quad : \text{EndFor} \quad : \text{Disp } i \]

**Note:** The `For` command automatically increments the counter variable so that the program can exit the loop after a certain number of repetitions.

At the end of the loop (`EndFor`), program control jumps back to the `For` command, where variable is incremented and compared to `end`.

For example:

```
:For i,0,5,1
  :Disp I
  :EndFor
:Disp i
```

1. Displays 0, 1, 2, 3, 4, and 5.
2. Displays 6. When `variable` increments to 6, the loop is not executed.

**Note:** You can declare the counter variable as local if it does not need to be saved after the program stops.
While...EndWhile Loops

A While...EndWhile loop repeats a block of commands as long as a specified condition is true. The syntax of the While command is:

\[
\text{While } \text{condition}
\]

When While is executed, the condition is evaluated. If condition is true, the loop is executed; otherwise, program control jumps to the command following EndWhile.

\[
\begin{aligned}
\text{x} & \geq 5 \\
\text{x} & < 5 \\
\text{While} & \text{ } \text{x} < 5 \\
\text{--------} \\
\text{--------} \\
\text{EndWhile} \\
\text{--------}
\end{aligned}
\]

Note: The While command does not automatically change the condition. You must include commands that allow the program to exit the loop.

At the end of the loop (EndWhile), program control jumps back to the While command, where condition is re-evaluated.

To execute the loop the first time, the condition must initially be true.

- Any variables referenced in the condition must be set before the While command. (You can build the values into the program or prompt the user to enter the values.)
- The loop must contain commands that change the values in the condition, eventually causing it to be false. Otherwise, the condition is always true and the program cannot exit the loop (called an infinite loop).
For example:

1. :0→x
   :While x<5
2. : Disp x
3. : x+1→x
   : EndWhile
4. : Disp x

1. Initially sets x.
2. Displays 0, 1, 2, 3, and 4.
3. Increments x.
4. Displays 5. When x increments to 5, the loop is not executed.

**Loop...EndLoop Loops**

A **Loop...EndLoop** creates an infinite loop, which is repeated endlessly. The **Loop** command does not have any arguments.

:Loop
  :  __________
  :  __________
  : EndLoop
  : __________
Typically, the loop contains commands that let the program exit from the loop. Commonly used commands are: **If**, **Exit**, **Goto**, and **Lbl** (label). For example:

```plaintext
:0→x
:Loop
:  Disp x
:  x+1→x
① :  If x>5
:     Exit
:     EndLoop
② :Disp x
```

① An **If** command checks the condition.
② Exits the loop and jumps to here when x increments to 6.

**Note:** The **Exit** command exits from the current loop.

In this example, the **If** command can be anywhere in the loop.

<table>
<thead>
<tr>
<th>When the <strong>If</strong> command is:</th>
<th>The loop is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the beginning of the loop</td>
<td>Executed only if the condition is true.</td>
</tr>
<tr>
<td>At the end of the loop</td>
<td>Executed at least once and repeated only if the condition is true.</td>
</tr>
</tbody>
</table>

The **If** command could also use a **Goto** command to transfer program control to a specified **Lbl** (label) command.
Repeating a Loop Immediately

The **Cycle** command immediately transfers program control to the next iteration of a loop (before the current iteration is complete). This command works with **For...EndFor**, **While...EndWhile**, and **Loop...EndLoop**.

**Lbl and Goto Loops**

Although the **Lbl** (label) and **Goto** commands are not strictly loop commands, they can be used to create an infinite loop. For example:

```
:Lbl START
:  --------
:  --------
:Goto START
:--------
```

As with **Loop...EndLoop**, the loop should contain commands that let the program exit from the loop.

**Configuring the TI-89 Titanium**

Programs can contain commands that change the configuration of the calculator. Because mode changes are particularly useful, the Program Editor’s **Mode** toolbar menu makes it easy to enter the correct syntax for the **setMode** command.
### Configuration Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getConfg</td>
<td>Returns a list of calculator characteristics.</td>
</tr>
<tr>
<td>getFold</td>
<td>Returns the name of the current folder.</td>
</tr>
<tr>
<td>getMode</td>
<td>Returns the current setting for a specified mode.</td>
</tr>
<tr>
<td>getUnits</td>
<td>Returns a list of default units.</td>
</tr>
<tr>
<td>setFold</td>
<td>Sets the current folder.</td>
</tr>
<tr>
<td>setGraph</td>
<td>Sets a specified graph format (Coordinates, Graph Order, etc.).</td>
</tr>
<tr>
<td>setMode</td>
<td>Sets any mode except Current Folder.</td>
</tr>
<tr>
<td>setTable</td>
<td>Sets a specified table setup parameter (tblStart, Δtbl, etc.)</td>
</tr>
<tr>
<td>setUnits</td>
<td>Sets default units for displayed results.</td>
</tr>
<tr>
<td>switch</td>
<td>Sets the active window in a split screen, or returns the number of the active window.</td>
</tr>
</tbody>
</table>

Note: The parameter/mode strings used in the setMode(), getMode(), setGraph(), and setTable() functions do not translate into other languages when used in a program. See the Technical Reference module.

### Entering the SetMode Command

In the Program Editor:
1. Position the cursor where you want to insert the setMode command.

2. Press: \( 2\text{nd} \ [F6] \) to display a list of modes.

   **Note:** The Mode menu does not let you set the Current Folder mode. To set this mode, use the setFold command.

3. Select a mode to display a menu of its valid settings.
4. Select a setting.

   The correct syntax is 
   
   \[
   \text{:setMode("Graph","FUNCTION")}
   \]

   inserted into your program.

---

### Getting Input from the User and Displaying Output

Although values can be built into a program (or stored to variables in advance), a program can prompt the user to enter information while the program is running. Likewise, a program can display information such as the result of a calculation.

---

### F3 I/O Toolbar Menu

To enter most of the commonly used input/output commands, use the Program Editor’s [F3] I/O toolbar menu.
To see a submenu that lists additional commands, select **1:Dialog**.

### Input Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>getKey</strong></td>
<td>Returns the key code of the next key pressed. See the <em>Technical Reference</em> module for a listing of key codes.</td>
</tr>
<tr>
<td><strong>Input</strong></td>
<td>Prompts the user to enter an expression. The expression is treated according to how it is entered. For example:</td>
</tr>
<tr>
<td></td>
<td>• A numeric expression is treated as an expression.</td>
</tr>
<tr>
<td></td>
<td>• An expression enclosed in &quot;quotes&quot; is treated as a string.</td>
</tr>
<tr>
<td></td>
<td><strong>Input</strong> can also display the Graph screen and let the user update the variables xc and yc (rc and θc in polar mode) by positioning the graph cursor.</td>
</tr>
<tr>
<td><strong>InputStr</strong></td>
<td>Prompts the user to enter an expression. The expression is always treated as a string; the user does not need to enclose the expression in &quot;quotes&quot;.</td>
</tr>
<tr>
<td><strong>PopUp</strong></td>
<td>Displays a pop-up menu box and lets the user select an item.</td>
</tr>
<tr>
<td><strong>Prompt</strong></td>
<td>Prompts the user to enter a series of expressions. As with <strong>Input</strong>, each expression is treated according to how it is entered.</td>
</tr>
</tbody>
</table>
**Request**

Displays a dialog box that prompts the user to enter an expression. **Request** always treats the entered expression as a string.

**Note:** String input cannot be used in a calculation. To convert a string to a numeric expression, use the `expr` command.

### Output Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ClrIO</strong></td>
<td>Clears the Program I/O screen.</td>
</tr>
<tr>
<td><strong>Disp</strong></td>
<td>Displays an expression or string on the Program I/O screen. <strong>Disp</strong> can also display the current contents of the Program I/O screen without displaying additional information.</td>
</tr>
<tr>
<td><strong>DispG</strong></td>
<td>Displays the current contents of the Graph screen.</td>
</tr>
<tr>
<td><strong>DispHome</strong></td>
<td>Displays the current contents of the Home screen.</td>
</tr>
<tr>
<td><strong>DispTbl</strong></td>
<td>Displays the current contents of the Table screen.</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>Displays an expression or string starting at specified coordinates on the Program I/O screen.</td>
</tr>
<tr>
<td><strong>Format</strong></td>
<td>Formats the way in which numeric information is displayed.</td>
</tr>
<tr>
<td><strong>Pause</strong></td>
<td>Suspends program execution until the user presses [ENTER]. Optionally, you can display an expression during the pause. A pause lets users read your output and decide when they are ready to continue.</td>
</tr>
</tbody>
</table>
• In a program, simply performing a calculation does not display the result. You must use an output command.

• After `Disp` and `Output`, the program immediately continues. You may want to add a `Pause` command.

### Graphical User Interface Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dialog...</strong></td>
<td>Defines a program block (consisting of <strong>Title</strong>, <strong>Request</strong>, etc., commands) that displays a dialog box.</td>
</tr>
<tr>
<td><strong>EndDlog</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Toolbar...</strong></td>
<td>Defines a program block (consisting of <strong>Title</strong>, <strong>Item</strong>, etc., commands) that replaces the toolbar menus. The redefined toolbar is in effect only while the program is running and only until the user selects an item. Then the original toolbar is redisplayed.</td>
</tr>
<tr>
<td><strong>EndTbar</strong></td>
<td></td>
</tr>
<tr>
<td><strong>CustmOn...</strong></td>
<td>Activates or removes a custom toolbar.</td>
</tr>
<tr>
<td><strong>CustmOff</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Custom...</strong></td>
<td>Defines a program block that displays a custom toolbar when the user presses <code>2nd [CATALOG]</code>. That toolbar remains in effect until the user presses <code>2nd [CATALOG]</code> again or changes applications.</td>
</tr>
<tr>
<td><strong>EndCustm</strong></td>
<td></td>
</tr>
</tbody>
</table>
### Command Description

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DropDown</td>
<td>Displays a drop-down menu within a dialog box.</td>
</tr>
<tr>
<td>Item</td>
<td>Displays a menu item for a redefined toolbar.</td>
</tr>
<tr>
<td>Request</td>
<td>Creates an input box within a dialog box.</td>
</tr>
<tr>
<td>Text</td>
<td>Displays a character string within a dialog box.</td>
</tr>
<tr>
<td>Title</td>
<td>Displays the title of a dialog box or a menu title within a toolbar.</td>
</tr>
</tbody>
</table>

### Notes:

- When you run a program that sets up a custom toolbar, that toolbar is still available even after the program has stopped.

- **Request** and **Text** are stand-alone commands that can also be used outside of a dialog box or toolbar program block.

### Creating a Custom Menu

The custom menu feature lets you create your own toolbar menu. A custom menu can contain any available function, instruction, or set of characters. The calculator has a default custom menu that you can modify or redefine.
Turning the Custom Menu On and Off

When you create a custom menu, you can let the user turn it on and off manually, or you can let a program turn it on and off automatically.

To: Do this:
---
Turn on the custom menu

From the Home screen or any other application:
- Press 2nd [CATALOG].

From the Home screen or a program:
- Execute the **CustmOn** command.

Turn off the custom menu

From any application:
- Press 2nd [CATALOG] again.
- or –
- Go to a different application.

Using the default custom menu on the Home screen:
1. Select the **Tools** menu:
   - 2nd [F7] Select 3: **CustmOff**.
   - This pastes **CustmOff** in the entry line.
2. Press [ENTER].
   - You can also use **CustmOff** in a program.

**Note:** When the custom menu is turned on, it replaces the normal toolbar menu. Unless a different custom menu has been created, the default custom menu is displayed.
Defining a Custom Menu

To create a custom menu, use the following general structure.

```
Custom
  : Title title of F1 menu:
    : Item item 1
    : Item item 2
    : ...
  : Title title of F2 menu
  : ...
  : Title title of F3 menu
  : ...
```

**Note:** When the user selects a menu item, the text defined by that `Item` command is pasted to the current cursor location.

For example:

```
:Custom
:Title "Vars"
:Item "L1":Item "M1":Item "Prgm1":Item "Func1":Item "Data1"
:Item "Text1":Item "Pic1":Item "GDB1":Item "Str1"

① :Title "f(x)"
① :Item "f(x)":Item "g(x)":Item "f(x,y)":Item "g(x,y)"
① :Item "f(x+h)":Item "Define f(x) ="

:Title "Solve"
:Item "Solve(":Item " and ":Item "{x,y}" 
:Item "Solve( and ,{x,y})"
```
Note: The following may be slightly different than the default custom menu on your calculator.

Note: See how \(_\circ{\text{C}}\) and \(_\circ{\text{F}}\) display as °C and °F in the menu. Similarly, see the international accented characters.

To modify the default custom menu, use 3:Restore custom default (as described below) to get the commands for the default menu. Copy those commands, use the Program
Editor to create a new program, and paste them into the blank program. Then modify the commands as necessary.

**Note:** This inserts all the commands on a single line. You do not need to split them into separate lines.

You can create and use only one custom menu at a time. If you need more, write a separate program for each custom menu. Then run the program for the menu you need.

**Restoring the Default Custom Menu**

To restore the default:

1. From the Home screen’s normal menu (not the custom menu), select **Clean Up**: 
   2nd [F6]

2. Select 3: **Restore custom default**.
   
   This pastes the commands used to create the default menu into the entry line.

3. Press **ENTER** to execute the commands and restore the default.

When you restore the default, any previous custom menu is erased. If the previous menu was created with a program, you can run the program again if you want to reuse the menu later.
Creating a Table or Graph

To create a table or a graph based on one or more functions or equations, use the commands listed in this section.

**Table Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DispTbl</td>
<td>Displays the current contents of the Table screen.</td>
</tr>
<tr>
<td>setTable</td>
<td>Sets the Graph $\leftrightarrow$ Table or Independent table parameters.</td>
</tr>
<tr>
<td></td>
<td>(To set the other two table parameters, you can store the applicable values</td>
</tr>
<tr>
<td></td>
<td>to the $\texttt{tblStart}$ and $\Delta \texttt{tbl}$ system variables.)</td>
</tr>
<tr>
<td>Table</td>
<td>Builds and displays a table based on one or more expressions or functions.</td>
</tr>
</tbody>
</table>

**Graphing Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClrGraph</td>
<td>Erases any functions or expressions that were graphed with the $\texttt{Graph}$ command.</td>
</tr>
<tr>
<td>Define</td>
<td>Creates a user-defined function.</td>
</tr>
<tr>
<td>DispG</td>
<td>Displays the current contents of the Graph screen.</td>
</tr>
<tr>
<td>FnOff</td>
<td>Deselects all (or only specified) $Y=\text{ functions}$.</td>
</tr>
<tr>
<td>FnOn</td>
<td>Selects all (or only specified) $Y=\text{ functions}$.</td>
</tr>
</tbody>
</table>
Graphs one or more specified expressions, using the current graphing mode.

Displays the Graph screen and lets the user update the variables xc and yc (rc and θc in polar mode) by positioning the graph cursor.

Creates a new stat plot definition.

Deselects all (or only specified) stat data plots.

Selects all (or only specified) stat data plots.

Changes settings for the various graph formats (Coordinates, Graph Order, etc.).

Sets the Graph mode, as well as other modes.

Sets the display style for a function.

Lets a program trace a graph.

Perform all of the **Zoom** operations that are available from the `F2` toolbar menu on the Y= Editor, Window Editor, and Graph screen.

**Note:** More information is available about using **setMode**.

### Graph Picture and Database Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AndPic</td>
<td>Displays the Graph screen and superimposes a stored graph picture by using <strong>AND</strong> logic.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>CyclePic</td>
<td>Animates a series of stored graph pictures.</td>
</tr>
<tr>
<td>NewPic</td>
<td>Creates a graph picture variable based on a matrix.</td>
</tr>
<tr>
<td>RclGDB</td>
<td>Restores all settings stored in a graph database.</td>
</tr>
<tr>
<td>RclPic</td>
<td>Displays the Graph screen and superimposes a stored graph picture by using OR logic.</td>
</tr>
<tr>
<td>RplcPic</td>
<td>Clears the Graph screen and displays a stored graph picture.</td>
</tr>
<tr>
<td>StoGDB</td>
<td>Stores the current graph settings to a graph database variable.</td>
</tr>
<tr>
<td>StoPic</td>
<td>Copies the Graph screen (or a specified rectangular portion) to a graph picture variable.</td>
</tr>
<tr>
<td>XorPic</td>
<td>Displays the Graph screen and superimposes a stored graph picture by using XOR logic.</td>
</tr>
</tbody>
</table>

**Note:** For information about graph pictures and databases, also refer to *Additional Graphing Topics*.

**Drawing on the Graph Screen**

To create a drawing object on the Graph screen, use the commands listed in this section.

**Pixel vs. Point Coordinates**

When drawing an object, you can use either of two coordinate systems to specify a location on the screen.
• **Pixel coordinates** — Refer to the pixels that physically make up the screen. These are independent of the viewing window because the screen is always: 159 (0 to 158) pixels wide and 77 (0 to 76) pixels tall.

• **Point coordinates** — Refer to the coordinates in effect for the current viewing window (as defined in the Window Editor).

<table>
<thead>
<tr>
<th>Pixel coordinates (independent of viewing window)</th>
<th>Point coordinates (for standard viewing window)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Pixel coordinates" /></td>
<td><img src="image2" alt="Point coordinates" /></td>
</tr>
</tbody>
</table>

**Note:** For information about pixel coordinates in split screens, refer to the Data/Matrix Editor module.

Many drawing commands have two forms: one for pixel coordinates and one for point coordinates.

**Note:** Pixel commands start with **Pxl**, such as **PxlChg**.

### Erasing Drawn Objects

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ClrDraw</strong></td>
<td>Erases all drawn objects from the Graph screen.</td>
</tr>
</tbody>
</table>
## Drawing a Point or Pixel

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PtChg or PxlChg</td>
<td>Toggles (inverts) a pixel at the specified coordinates. PtChg, which uses point coordinates, affects the pixel closest to the specified point. If the pixel is off, it is turned on. If the pixel is on, it is turned off.</td>
</tr>
<tr>
<td>PtOff or PxlOff</td>
<td>Turns off (erases) a pixel at the specified coordinates. PtOff, which uses point coordinates, affects the pixel closest to the specified point.</td>
</tr>
<tr>
<td>PtOn or PxlOn</td>
<td>Turns on (displays) a pixel at the specified coordinates. PtOn, which uses point coordinates, affects the pixel closest to the specified point.</td>
</tr>
<tr>
<td>PtTest or PxlTest</td>
<td>Returns true or false to indicate if the specified coordinate is on or off, respectively.</td>
</tr>
<tr>
<td>PtText or PxlText</td>
<td>Displays a character string at the specified coordinates.</td>
</tr>
</tbody>
</table>

## Drawing Lines and Circles

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circle or PxlCrcI</td>
<td>Draws, erases, or inverts a circle with a specified center and radius.</td>
</tr>
<tr>
<td>DrawSlp</td>
<td>Draws a line with a specified slope through a specified point.</td>
</tr>
<tr>
<td>Line or PxlLine</td>
<td>Draws, erases, or inverts a line between two sets of coordinates.</td>
</tr>
</tbody>
</table>
### Drawing Expressions

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LineHorz or PxlHorz</td>
<td>Draws, erases, or inverts a horizontal line at a specified row coordinate.</td>
</tr>
<tr>
<td>LineTan</td>
<td>Draws a tangent line for a specified expression at a specified point. (This draws the tangent line only, not the expression.)</td>
</tr>
<tr>
<td>LineVert or PxlVert</td>
<td>Draws, erases, or inverts a vertical line at a specified column coordinate.</td>
</tr>
<tr>
<td>DrawFunc</td>
<td>Draws a specified expression.</td>
</tr>
<tr>
<td>DrawInv</td>
<td>Draws the inverse of a specified expression.</td>
</tr>
<tr>
<td>DrawParm</td>
<td>Draws a parametric equation using specified expressions as its x and y components.</td>
</tr>
<tr>
<td>DrawPol</td>
<td>Draws a specified polar expression.</td>
</tr>
<tr>
<td>DrwCtour</td>
<td>Draws contours in 3D graphing mode.</td>
</tr>
<tr>
<td>Shade</td>
<td>Draws two expressions and shades the areas where $expression1 &lt; expression2$.</td>
</tr>
</tbody>
</table>
Accessing Another TI-89 Titanium, a CBL 2, or a CBR

If you link two graphing calculators (described in the Connectivity module), programs on both units can transmit variables between them. If you link a TI-89 Titanium to a Calculator-Based Laboratory™ (CBL 2™) or a Calculator-Based Ranger™ (CBR™), a program on the TI-89 Titanium can access the CBL 2 or CBR.

F3 I/O Toolbar Menu

Use the Program Editor’s F3 I/O toolbar menu to enter the commands in this section.

1. Press F3 and select 8:Link.
2. Select a command.
Accessing Another TI-89 Titanium

When two calculators are linked, one acts as a receiving unit and the other as a sending unit.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| GetCalc  | Executed on the receiving unit. Sets up the unit to receive a variable via the I/O port.  
  • After the receiving unit executes **GetCalc**, the sending unit must execute **SendCalc**.  
  • After the sending unit executes **SendCalc**, the sent variable is stored on the receiving unit (in the variable name specified by **GetCalc**). |
| SendCalc | Executed on the sending unit. Sends a variable to the receiving unit via the I/O port.  
  • Before the sending unit executes **SendCalc**, the receiving unit must execute **GetCalc**. |

**Note:** For a sample program that synchronizes the receiving and sending units so that **GetCalc** and **SendCalc** are executed in the proper sequence, refer to “Transmitting Variables under Program Control” in Connectivity.

Accessing a CBL 2 or CBR

For additional information, refer to the manual that comes with the CBL 2 or CBR unit.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get</td>
<td>Gets a variable from an attached CBL 2 or CBR and stores it in the graphing calculator.</td>
</tr>
</tbody>
</table>
Debugging Programs and Handling Errors

After you write a program, you can use several techniques to find and correct errors. You can also build an error-handling command into the program itself.

Run-Time Errors

The first step in debugging your program is to run it. The graphing calculator automatically checks each executed command for syntax errors. If there is an error, a message indicates the nature of the error.

- To display the program in the Program Editor, press [ENTER]. The cursor appears in the approximate area of the error.

- To cancel program execution and return to the Home screen, press [ESC].

If your program allows the user to select from several options, be sure to run the program and test each option.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send</td>
<td>Sends a list variable from the graphing calculator to the CBL 2 or CBR.</td>
</tr>
</tbody>
</table>
Debugging Techniques

Run-time error messages can locate syntax errors but not errors in program logic. The following techniques may be useful.

- During testing, do not use local variables so that you can check the variable values after the program stops. When the program is debugged, declare the applicable variables as local.
- Within a program, temporarily insert `Disp` and `Pause` commands to display the values of critical variables.
  - `Disp` and `Pause` cannot be used in a user-defined function. To temporarily change the function into a program, change `Func` and `EndFunc` to `Prgm` and `EndPrgm`. Use `Disp` and `Pause` to debug the program. Then remove `Disp` and `Pause` and change the program back into a function.
- To confirm that a loop is executed the correct number of times, display the counter variable or the values in the conditional test.
- To confirm that a subroutine is executed, display messages such as `Entering subroutine` and `Exiting subroutine` at the beginning and end of the subroutine.

Error-Handling Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Try...EndTry</td>
<td>Defines a program block that lets the program execute a command and, if necessary, recover from an error generated by that command.</td>
</tr>
<tr>
<td>ClrErr</td>
<td>Clears the error status and sets the error number in system variable Errornum to zero.</td>
</tr>
<tr>
<td>PassErr</td>
<td>Passes an error to the next level of the Try...EndTry block.</td>
</tr>
</tbody>
</table>
Example: Using Alternative Approaches

The example in the *Previews* module shows a program that prompts the user to enter an integer, sums all integers from 1 to the entered integer, and displays the result. This section gives several approaches that you can use to achieve the same goal.

**Example 1**

This example uses **InputStr** for input, a **While...EndWhile** loop to calculate the result, and **Text** to display the result.

```
:prog1()
 :Prgm
  1 :InputStr "Enter an integer",n
  2 :expr(n)→n
      :0→temp:1→I
  3 :While i≤n
     :  temp+i→temp
     :  i+1→I
  3 :EndWhile
  4 :Text "The answer is "&string(temp)
     :EndPrgm
```

1. Prompts for input on Program I/O screen.
2. Converts string entered with **InputStr** to an expression.
3. Loop calculation.
4. Displays output in a dialog box.
**Example 2**

This example uses **Prompt** for input, **Lbl**, and **Goto** to create a loop, and **Disp** to display the result.

```
:prog2()
:Prgm
① :Prompt n
   :0→temp:1→I
② :Lbl top
   . : temp+i→temp
   . : i+1→I
   . : If i≤n
   ③ :   Goto top
③ :Disp temp
   :EndPrgm
```

① Prompts for input on Program I/O screen.
② Loop calculation.
③ Displays output on Program I/O screen.

**Note:** Because **Prompt** returns \( n \) as a number, you do not need to use **expr** to convert \( n \).
Example 3

This example uses **Dialog...EndDlog** to create dialog boxes for input and output. It uses **Loop...EndLoop** to calculate the result.

```
:prog3()
  :Prgm
  :Dialog
  :   Title "Enter an integer"
  :   Request "Integer",n
  :EndDlog
  :expr(n)→n
  :0→temp:0→I
  :Loop
  :   temp+i→temp
  :   i+1→I
  :   If i>n
  :     Exit
  :EndLoop
  :Dialog
  :   Title "The answer is"
  :   Text string(temp)
  :EndDlog
  :EndPrgm
```

1. Defines a dialog box for input.
2. Converts string entered with **Request** to an expression.
3. Loop calculation.
4. Defines a dialog box for output.
Example 4

This example uses built-in functions to calculate the result without using a loop.

```
: prog4()
: Prgm
1: Input "Enter an integer", n
2: sum(seq(i,i,1,n)) → temp
3: Disp temp
: EndPrgm
```

1: Prompts for input on Program I/O.
2: Calculates sum.
3: Displays output on Program I/O screen.

**Note:** Because `Input` returns `n` as a number, you do not need to use `expr` to convert `n`.

<table>
<thead>
<tr>
<th>Function</th>
<th>Used in this example to:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>seq</code></td>
<td>Generate the sequence of integers from 1 to <code>n</code>.</td>
</tr>
</tbody>
</table>

```
seq(expression, var, low, high [,step])
```

1: expression used to generate the sequence
2: variable that will be incremented
3: initial and final values of `var`
4: increment for `var`; if omitted, uses 1

`sum` Sum the integers in the list generated by `seq`. 
Assembly-Language Programs

You can run programs written for the TI-89 Titanium in assembly language. Typically, assembly-language programs run much faster and provide greater control than the keystroke programs that you write with the built-in Program Editor.

Where to Get Assembly-Language Programs

Assembly-language programs, as well as keystroke programs, are available on the Texas Instruments web site at education.ti.com.

The programs available from this site provide additional functions or features that are not built into the TI-89 Titanium. Check the Texas Instruments web site for up-to-date information.

After downloading a program from the web to your computer, use a USB cable or TI-GRAPH LINK™ computer-to-calculator cable and TI Connect software to send the program to your TI-89 Titanium.

For Flash App installation instructions, see education.ti.com/guides.

Note about TI-GRAPH LINK

If you have a TI-GRAPH LINK™ computer-to-calculator cable and software for the TI-89 or TI-92 Plus, be aware that the TI-GRAPH LINK software is not compatible with the TI-89 Titanium. The cable, however, works with all units. Use TI Connect software on your computer.
You can purchase computer-to-calculator and unit-to-unit cables from the TI Online Store at education.ti.com/buy.

### Running an Assembly-Language Program

After a TI-89 Titanium assembly-language program is stored on your unit, you can run the program from the Home screen just as you would any other program.

- If the program requires one or more arguments, type them within the ( ). Refer to the program’s documentation to find out about required arguments.

```plaintext
prgmName(
MAIN RAD AUTO FUNC 0/30
```

- If the program is not in the current folder, be sure to specify the pathname.

You can call an assembly-language program from another program as a subroutine, delete it, or use it the same as any other program.

### Shortcuts to Run a Program

On the Home screen, you can use keyboard shortcuts to run up to six user-defined or assembly-language programs. However, the programs must have the following names.

<table>
<thead>
<tr>
<th>On Home screen, press:</th>
<th>To run a program, if any, named:</th>
</tr>
</thead>
<tbody>
<tr>
<td>♠ 1</td>
<td>kbdprgm1( )</td>
</tr>
<tr>
<td>♠ 6</td>
<td>kbdprgm6( )</td>
</tr>
</tbody>
</table>
The programs must be stored in the **MAIN** folder. Also, you cannot use a shortcut to run a program that requires an argument.

If you have a program with a different name and you would like to run it with a keyboard shortcut, copy or rename the existing program to `kbdprgm1()`, etc.

**You Cannot Edit an Assembly-Language Program**

You cannot use your TI-89 Titanium to edit an assembly-language program. The built-in Program Editor will not open assembly-language programs.

**Displaying a List of Assembly-Language Programs**

To list the assembly-language programs stored in memory:

1. Display the **VAR-LINK** screen (`2nd` [VAR-LINK]).
2. Press `F2` View.
3. Select the applicable folder (or All folders) and set **Var Type = Assembly**.
4. Press `ENTER` to display the list of assembly-language programs.

**Note:** Assembly-language programs have an **ASM** data type.
For Information about Writing an Assembly-Language Program

The information required to teach a novice programmer how to write an assembly-language program is beyond the scope of this book. However, if you have a working knowledge of assembly language, please check the Texas Instruments web site (education.ti.com) for specific information about how to access TI-89 Titanium features.

The graphing calculator also includes an Exec command that executes a string consisting of a series of Motorola 68000 op-codes. These codes act as another form of an assembly-language program. Check the Texas Instruments web site for available information.

**Note:** You must use a computer to write assembly-language programs. You cannot create assembly-language programs from the calculator keyboard.

**Warning:** Exec gives you access to the full power of the microprocessor. Please be aware that you can easily make a mistake that locks up the calculator and causes you to lose your data. We suggest you make a backup of the calculator contents before attempting to use the Exec command.
Starting a Text Editor Session

Each time you start the Text Editor, you can start a new text session, resume the current session (the session that was displayed the last time you used the Text Editor), or open a previous session.

Starting a New Session

1. Press \texttt{APPS} and then select the Text Editor icon. Press \texttt{ENTER}.

2. Select \texttt{3:New}.

   The \texttt{NEW} dialog box is displayed.

3. Specify a folder and text variable that you want to use to store the new session.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Automatically set as \texttt{Text} and cannot be changed.</td>
</tr>
<tr>
<td>Folder</td>
<td>Shows the folder in which the text variable will be stored. For information about folders, refer to the \textit{Calculator Home Screen module}. To use a different folder, press \texttt{①} to display a menu of existing folders. Then select a folder.</td>
</tr>
</tbody>
</table>
4. Press \[ \text{ENTER} \] (after typing in an input box such as \textbf{Variable}, you must press \[ \text{ENTER} \] twice) to display an empty Text Editor screen.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Type a variable name. If you specify a variable that already exists, an error message will be displayed when you press [ \text{ENTER} ]. When you press [ \text{ESC} ] or [ \text{ENTER} ] to acknowledge the error, the \textbf{NEW} dialog box is redisplayed.</td>
</tr>
</tbody>
</table>

A colon marks the beginning of a paragraph.

The blinking cursor shows where typed text will appear.

You can now use the Text Editor as described in the remaining sections of this module.

\textbf{Note:} Your session is saved automatically as you type. You do not need to save a session manually before leaving the Text Editor, starting a new session, or opening a previous one.

\textbf{Resuming the Current Session}

You can leave the Text Editor and go to another application at any time. To return to the session that was displayed when you left the Text Editor, launch Text Editor again and select 1:Current.
Starting a New Session from the Text Editor

To leave the current Text Editor session and start a new one:

1. Press **F1** and select **3:New**.
2. Specify a folder and text variable for the new session.
3. Press **ENTER** twice.

Opening a Previous Session

You can open a previous Text Editor session at any time.

1. From within the Text Editor, press **F1** and select **1:Open**.
   — or —
   From any application, launch Text Editor again and select **2:Open**.
2. Select the applicable folder and text variable.
3. Press **ENTER**.

Note: By default, **Variable** shows the first existing text variable in alphabetic order.
Copying a Session

In some cases, you may want to copy a session so that you can edit the copy while retaining the original.

1. Display the session you want to copy.
2. Press \[F1\] and select **2:Save Copy As**.
3. Specify the folder and text variable for the copied session.
4. Press \[ENTER\] twice.

Note about Deleting a Session

Because all Text Editor sessions are saved automatically, you can accumulate quite a few previous sessions, which take up memory storage space.

To delete a session, use the VAR-LINK screen (\[2nd\] \[VAR-LINK\]) to delete that session’s text variable. For information about VAR-LINK, refer to *Memory and Variable Management*.

Entering and Editing Text

After beginning a Text Editor session, you can enter and edit text. In general, use the same techniques that you have already used to enter and edit information on the Home screen’s entry line.
Typing Text

When you create a new Text Editor session, you see an empty screen. When you open a previous session or return to the current session, you see the existing text for that session.

All text paragraphs begin with a space and a colon.

The beginning space is used in command scripts and lab reports.

You do not need to press \texttt{\textasciitilde} at the end of each line. At the end of a line, the next character you type wraps to the next line. Press \texttt{\textasciitilde} only when you want to start a new paragraph.

As you reach the bottom of the screen, previous lines scroll off the top of the screen.

Using a USB cable and TI Connect\textsuperscript{TM} software with the TI-89 Titanium, you can use the computer keyboard to type a text file and then send that file to the TI-89 Titanium. This is useful if you need to create a lengthy text file.

For information about obtaining cables or updated TI Connect\textsuperscript{TM} software, check the TI web site at \texttt{education.ti.com}, or contact Texas Instruments at TI-Cares\textsuperscript{TM}.

Notes:

\begin{itemize}
  \item Use the cursor pad to scroll through a session or position the text cursor.
\end{itemize}
• Press \textbf{2nd} \circledsmallarrows or \textbf{2nd} \circlesrightarrows to scroll up or down one screen at a time, and \sphericalangle or \textbf{\textdagger} \circledsmallarrows or \textbf{\textdagger} \circlesrightarrows to go to the top or bottom of the text session.

**Typing Alphabetic Characters**

<table>
<thead>
<tr>
<th>To:</th>
<th>Press:</th>
</tr>
</thead>
</table>
| Type a single lowercase alpha character. | \begin{array}{c}	ext{Press:} \\
\text{\text{[alpha] and then the letter key (status line shows \text{[1]} 
}\end{array} |

| | |
| Type a single uppercase alpha character. | \begin{array}{c}	ext{Press:} \\
\text{\text{[1] and then the letter key (status line shows \text{[1]} 
}\end{array} |

| Type a space. | \begin{array}{c}	ext{Press:} \\
\text{\text{[alpha [~] (alpha function of the \text{[\textdagger]} key)} 
}\end{array} |

| Turn on lowercase alpha-lock. | \begin{array}{c}	ext{Press:} \\
\text{\text{[2nd [a-lock] (status line shows \text{[2nd [a-lock]} 
}\end{array} |

| Turn on uppercase ALPHA-lock. | \begin{array}{c}	ext{Press:} \\
\text{\text{[1] [a-lock] (status line shows \text{[1] [a-lock]} 
}\end{array} |

| Turn off alpha-lock. | \begin{array}{c}	ext{Press:} \\
\text{\text{[alpha (turns off upper- and lowercase lock)} 
}\end{array} |

**Note:** On the TI-89 Titanium, you do not need \text{[alpha] or alpha-lock to type \text{x, y, z, or t}. But you must use \text{[1] or uppercase ALPHA-lock for \text{X, Y, Z, or T}. On the TI-89 Titanium, alpha-lock is always turned off when you change applications, such as going from the Text Editor to the Home screen.
On the TI-89 Titanium, while either type of alpha-lock is on:

- To type a period, comma, or other character that is the primary function of a key, you must turn alpha-lock off.
- To type a second function character such as `2nd [t]`, you do not need to turn alpha-lock off. After you type the character, alpha-lock remains on.

**Deleting Characters**

<table>
<thead>
<tr>
<th>To delete:</th>
<th>Press:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The character to the left of the cursor</td>
<td>← or [F1] 7</td>
</tr>
<tr>
<td>The character to the right of the cursor</td>
<td>[DELS] (same as [DELS] ←)</td>
</tr>
<tr>
<td>All characters to the right of the cursor through the end of the paragraph</td>
<td>CLEAR</td>
</tr>
<tr>
<td>All characters in the paragraph (regardless of the cursor’s position in that paragraph)</td>
<td>CLEAR CLEAR</td>
</tr>
</tbody>
</table>

**Note:** If there are no characters to the right of the cursor, CLEAR erases the entire paragraph.
Highlighting Text

To: Do this:

Highlight text  Move the cursor to the beginning or end of the text. Hold ▲ and press:
  •  ◀ or ▶ to highlight characters to the left or right of the cursor, respectively.
  •  ◄ or ◄ to highlight all characters up to the cursor position on the next or previous line, respectively.

Note: To remove highlighting without replacing or deleting, move the cursor.

Replacing or Deleting Highlighted Text

To: Do this:

Replace highlighted text  Type the new text.

Delete highlighted text  Press ◄.
Cutting, Copying, and Pasting Text

Cutting and copying both place highlighted text into the clipboard of the TI-89 Titanium. Cutting deletes the text from its current location (used to move text) and copying leaves the text.

1. Highlight the text you want to move or copy.

2. Press \( \text{F1} \).

3. Select the applicable menu item.
   - To move the text, select \( 4:\text{Cut} \).
   — or —
   - To copy the text, select \( 5:\text{Copy} \).

   **Note:** You can press:
   \( \text{[CUT]} \), \( \text{[COPY]} \), \( \text{[PASTE]} \)
to cut, copy, and paste without having to use the \( \text{F1} \) toolbar menu.

4. Move the text cursor to the location where you want to insert the text.

5. Press \( \text{F1} \) and then select \( 6:\text{Paste} \).

You can use this general procedure to cut, copy, and paste text:

- Within the same text session.
- From one text session to another. After cutting or copying text in one session, open the other session and then paste the text.
- From a text session to a different application. For example, you can paste the text into the Home screen’s entry line.
Finding Text

From the Text Editor:

1. Place the text cursor at any location preceding the text you want to search for. All searches start at the current cursor location.

2. Press \[F5\].

3. Type the search text.

   The search is not case sensitive. For example: CASE, case, and Case have the same effect.

   **Note:** The FIND dialog box retains the last search text you entered. You can type over it or edit it.

4. Press \[ENTER\] twice.

<table>
<thead>
<tr>
<th>If the search text is:</th>
<th>The cursor:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Found</td>
<td>Moves to beginning of the search text.</td>
</tr>
<tr>
<td>Not found</td>
<td>Does not move.</td>
</tr>
</tbody>
</table>
Inserting or Overtyping a Character

By default, the TI-89 Titanium is in insert mode. To toggle between insert and overtype mode, press [2nd] [INS].

<table>
<thead>
<tr>
<th>If the TI-89 Titanium is in:</th>
<th>The next character you type:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Insert mode</strong></td>
<td>Will be inserted at the cursor.</td>
</tr>
<tr>
<td>Thin cursor between characters</td>
<td></td>
</tr>
<tr>
<td><strong>OverType mode</strong></td>
<td>Will replace the highlighted character.</td>
</tr>
<tr>
<td>Cursor highlights a character</td>
<td></td>
</tr>
</tbody>
</table>

Note: Look at the shape of the cursor to see if you’re in insert or overtype mode.

Clearing the Text Editor

To erase all existing paragraphs and display an empty text screen, press [F1] and then select 8:Clear Editor.

Entering Special Characters

You can use the CHAR menu to select any special character from a list. You can also type certain commonly used characters from the keyboard. To see which characters are available from the keyboard, you can display a map that shows the characters and their corresponding keys.
Selecting Characters from the CHAR Menu

1. Press 2nd [CHAR].

2. Select the applicable category.
   A menu lists the characters in that category.

3. Select a character. You may need to scroll through the menu.

   **Note:** For accented characters, select International. Commonly used international characters are also available from the default custom menu (2nd [CUSTOM]).

Displaying the Keyboard Map

The keyboard map shows several shortcuts that let you enter certain special characters from the keyboard. It also shows some shortcuts for other calculator features.

The keyboard map does not display all available shortcuts. Refer to the inside front and the inside back covers of this guidebook for a complete list of shortcut keys.

To access the shortcuts, first press the 2nd key. Some special characters are marked on the keyboard, but most are not.

**On the TI-89 Titanium:**

- Press \( \downarrow \) to display the keyboard map.
- Press \[\text{ESC}\] to exit the map.

**TI-89 Titanium Keyboard map**

To access the TI-89 Titanium shortcuts, first press the \(\text{\textbullet}\) key.

**TI-89 Titanium keyboard map feature shortcuts:**

- **GREEK \(\text{\textbullet} \text{[ ]}\) —** Accesses the Greek character set (described later in this section).
- **SYSDATA \(\text{\textbullet} \text{[]}\) —** Copies the current graph coordinates to the system variable sysdata.
- **FMT \(\text{\textbullet} \text{[ ]}\) —** Displays the FORMATS dialog box.
- **KBDPRGM1 – 6\(\text{\textbullet} \text{[ ]} \text{1 through [ ]} \text{6}\) —** If you have user-defined or assembly-language programs named kbdprgm1() through kbdprgm6(), these shortcuts run the corresponding program.
- **OFF \(\text{\textbullet} \text{[OFF]}\) —** Similar to \[\text{2nd} \text{[OFF]}\] except:
  - You can use \(\text{\textbullet} \text{[OFF]}\) if an error message is displayed.
  - When you turn the TI-89 Titanium on again, it will be exactly as you left it.
• HOMEDATA (✓ (-)) — Copies the current graph coordinates to the Home screen’s history area.

**Typing Special Symbols from the Keyboard**

**Note:** To help you find the applicable keys, these maps show only the special symbols.

**On the TI-89 Titanium:**

- Press ✓ and then the key for the symbol. For example: ✓ × (times) displays &.

![Keyboard layout image](image)

These special symbols are not affected by whether Alpha-Lock is on or off.

**Typing Greek Letters from the Keyboard**

Press the key combination that accesses the Greek character set on your calculator. Then select the applicable alpha character on the keyboard to enter a Greek letter.
On the TI-89 Titanium:

- Press ◊ □ to access the Greek character set.

Note: If you press a key combination that does not access a Greek letter, you get the normal letter for that key. Your calculator does not display a map of Greek letters; the map shown here is for reference only.

Several keys let you access lowercase and uppercase Greek letters. For example:

On the TI-89 Titanium:

- Press ◊ □ to access the Greek character set.
- Press ◊ □ [alpha] + letter to access lowercase Greek letters. Example: ◊ □ [alpha] [W] displays ω
• Press $\diamond \Gamma \Uparrow$ + letter to access uppercase Greek letters. Example: $\diamond \Gamma \Uparrow$ [W] displays $\Omega$

The exact keys that you press on the TI-89 Titanium depend on whether alpha-lock is on or off. For example:

<table>
<thead>
<tr>
<th>On the TI-89 Titanium, if:</th>
<th>Then:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha-lock is off.</td>
<td>$\diamond \Gamma$ X or $\diamond \Gamma$ alpha X displays $\xi$. (&lt;sup&gt;alpha&lt;/sup&gt; is not required for X, Y, Z, or T.)</td>
</tr>
<tr>
<td></td>
<td>$\diamond \Gamma$ alpha W displays $\omega$.</td>
</tr>
<tr>
<td></td>
<td>$\diamond \Gamma \Uparrow$ W displays $\Omega$. (&lt;sup&gt;Uppercase alpha-lock ($\Uparrow$) is on.&lt;/sup&gt;)</td>
</tr>
</tbody>
</table>

| Lowercase alpha-lock (<sup>2nd a-lock</sup>) is on. | $\diamond \Gamma$ X displays $\xi$. |
|                                                      | $\diamond \Gamma$ W displays $\omega$. |
|                                                      | $\diamond \Gamma \Uparrow$ W displays $\Omega$. |

| Uppercase ALPHA-LOCK (<sup>Uppercase alpha-lock ($\Uparrow$) is on.</sup>) | $\diamond \Gamma$ X displays $\xi$. |
|                                                                         | $\diamond \Gamma$ W displays $\omega$. |
|                                                                         | $\diamond \Gamma \Uparrow$ W displays $\Omega$. |

**Important:** If you press $\textbf{alpha}$ on the TI-89 Titanium to access a Greek letter while alpha-lock is on, it turns alpha-lock off.

**For a List of All Special Characters**

For a list of all special characters, refer to the *Technical Reference* module.
Enter and Executing a Command Script

By using a command script, you can use the Text Editor to type a series of command lines that can be executed at any time on the Home screen. This lets you create interactive example scripts in which you predefine a series of commands and then execute them individually.

Inserting a Command Mark

In the Text Editor:

1. Place the cursor on the line for the command.

2. Press F2 to display the Command toolbar menu.


C is displayed at the beginning of the text line (to the left of the colon).

Note: This does not insert a new line for the command, it simply marks an existing line as a command line.
4. Type a command just as you would on the Home screen.

   The line can contain only the command, with no additional text.

   **Note:** You can mark a line as a command either before or after typing the command on that line.

   You can type multiple commands on the same line if you type a colon to separate the commands.

**Deleting a Command Mark**

This deletes only the C mark; it does not delete the command text itself.

1. Place the cursor anywhere on the marked line.
2. Press F2 and select **4:Clear command**.

**Executing a Command**

To execute a command, you must first mark the line with a C. If you execute a line that is not marked with C, it will be ignored.

1. Place the cursor anywhere on the command line.
The command is copied to the entry line on the Home screen and executed. The Home screen is displayed temporarily during execution, and then the Text Editor is redisplayed.

After execution, the cursor moves to the next line in the script so that you can continue to execute a series of commands.

**Note:** To examine the result on the Home screen, use a split screen or press `HOME`-

### Splitting the Text Editor/ Home Screen

With a split screen, you can view your command script and see the result of an executed command at the same time.

<table>
<thead>
<tr>
<th>To:</th>
<th>Press:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split the screen</td>
<td>F3 and select 1:Script view.</td>
</tr>
<tr>
<td>Return to a full screen</td>
<td>F3 and select 2:Clear split.</td>
</tr>
</tbody>
</table>

You can also use MODE to set up a split screen manually. However, F3 sets up a Text Editor/Home screen split much easier than MODE.

- The active application is indicated by a thick border. (By default, the Text Editor is the active application.)
- To switch between the Text Editor and the Home screen, press 2nd [APPs] (second function of APPS).
Creating a Script from Your Home Screen Entries

From the Home screen, you can save all the entries in the history area to a text variable. The entries are automatically saved in a script format so that you can open the text variable in the Text Editor and execute the entries as commands.

For information, refer to “Saving the Home Screen Entries as a Text Editor Script” in the Calculator Home Screen module.
Example

1. Type your script. Press \textbf{F2} and select \textbf{1:Command} to mark the command lines.

2. Press \textbf{F3} and select \textbf{1:Script view}.

3. Move the cursor to the first command line. Then press \textbf{F4} to execute the command.

   \textbf{Note:} Some commands take longer to execute. Wait until the \textbf{Busy} indicator disappears before pressing \textbf{F4} again.

4. Continue using \textbf{F4} to execute each command, but stop just before executing the Graph command.

5. Execute the Graph command.

   \textbf{Note:} In this example, the Graph command displays the Graph screen in place of the Home screen.

6. Press \textbf{F3} and select \textbf{2:Clear split} to return to a full screen Text Editor.
Displaying the Solver and Entering an Equation

After you display the Numeric Solver, start by entering the equation that you want to solve.

Displaying the Numeric Solver

To display the Numeric Solver, press \text{APPS} and then select the Numeric Solver icon. Press \text{ENTER}.

The Numeric Solver screen shows the last entered equation, if any.

Entering an Equation

On the \text{eqn:} line, type in your equation.

\begin{tabular}{|l|}
\hline
\textbf{You can:} & \textbf{For example:} \\
\hline
Type an equation directly. & \text{a}=(\text{m2}\text{−}\text{m1})/(\text{m2}+\text{m1})\text{g} \\
& \text{a+b=c+sin(d)} \\
\hline
\end{tabular}
<table>
<thead>
<tr>
<th>You can:</th>
<th>For example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refer to a function or equation defined elsewhere. <strong>Notes:</strong></td>
<td>Suppose you defined ( y_1(x) ) on either the:</td>
</tr>
</tbody>
</table>
| • Do not use system function names (such as \( y_1(x) \) or \( r_1(\theta) \)) as simple variables (\( y_1 \) or \( r_1 \)). | • \( Y= \) Editor:  
\[ y_1(x)=1.25x \cos(x) \]  
– or –  
• Home screen:  
**Define** \( y_1(x)=1.25x \cos(x) \) |
| • Be careful with implied multiplication. For example, \( a(m_2+m_1) \) is treated as a function reference, not as \( a*(m_2+m_1) \). | In the Numeric Solver, you then would enter:  
\( y_1(x)=0 \) or \( y_1(t)=0 \), etc.  
| | The argument does not have to match the one used to define the function or equation. |
| Type an expression without an = sign.  
**Note:** When you define the variables, you can either define \( \text{exp} \) or solve for it. | \( e+f-\ln(g) \)  
After you press [ENTER], the expression is set equal to a system variable called \( \text{exp} \) and entered as:  
\( \text{exp}=e+f-\ln(g) \) |
| Recall a previously entered equation or open a saved equation.  
**Note:** After you press [ENTER] the current equation is stored automatically to the system variable \( \text{eqn} \). | Refer to the applicable heading later in this section. |
Recalling Previously Entered Equations

Your most recently entered equations (up to 11 with the default setting) are retained in memory. To recall one of these equations:

1. From the Numeric Solver screen, press \( \text{F5} \).

   A dialog box displays the most recently entered equation.

2. Select an equation.
   - To select the displayed equation, press \( \text{ENTER} \).
   - To select a different equation, press \( \triangleright \) to display a list. Then select the one you want.

   **Note:** You can specify how many equations are retained. From the Numeric Solver, press \( \text{F1} \) and select 9:Format (or use \( \text{A} \ \text{D} \ \text{I} \)). Then select a number from 1 through 11.

3. Press \( \text{ENTER} \).
Saving Equations for Future Use

Because the number of equations that you can recall with \[F5\] Eqns is limited, a particular equation may not be retained indefinitely.

To store the current equation for future use, save it to a variable.

1. From the Numeric Solver screen, press \[F1\] and select **2:Save Copy As**.
2. Specify a folder and a variable name for the equation.
3. Press **ENTER** twice.

**Note:** An equation variable has an EXPR data type, as shown on the MEMORY and VAR-LINK screens.

Opening a Saved Equation

To open a previously saved equation variable:

1. From the Numeric Solver screen, press \[F1\] and select **1:Open**.
2. Select the applicable folder and equation variable.

3. Press ENTER.

Defining the Known Variables

After you type an equation in the Numeric Solver, enter the applicable values for all variables except the unknown variable.

Defining the List of Variables

After typing your equation on the eqn: line, press ENTER or \( \triangleright \).

The screen lists the variables in the order they appear in the equation. If a variable is already defined, its value is shown. You can edit these variable values.

Note: If an existing variable is locked or archived, you cannot edit its value.
Enter a number or expression for all variables except the one you want to solve for.

**Notes and Common Errors**

- **If you define a variable:**
  - In terms of another variable in the equation, that variable must be defined first.
  - In terms of another variable that is not in the equation, that variable must already have a value; it cannot be undefined.
  - As an expression, it is evaluated when you move the cursor off the line. The expression must evaluate to a real number.

- **If the equation contains a variable already defined in terms of other variables, those other variables are listed.**

**Note:** When you assign a value to a variable in the Numeric Solver, that variable is defined globally. It still exists after you leave the solver.

Since a is defined in terms of g, you must define g before a. When you move the cursor to another line, g/3 is evaluated.

If variable a was defined previously as b+c→a, then b and c are listed instead of a.
• If you refer to a previously defined function, any variables used as arguments in the function call are listed, not the variables used to define the function.

• If the equation contains a system variable (xmin, xmax, etc.), that variable is not listed. The solver uses the system variable’s existing value.

  **Note:** You cannot solve for a system variable other than exp. Also, if the equation contains a system variable, you cannot use F3 to graph.

• Although you can use a system variable in the equation, an error occurs if you use F3 to graph the solution.
• If you see the error shown to the right, delete the entered variable value. Then edit the equation to use a different variable.

Note: This error occurs if you use a reserved name incorrectly or refer to an undefined system function as a simple variable without parentheses.

Editing the Equation

In the Numeric Solver, press \( \text{C} \) until the cursor is on the equation. The screen automatically changes to show only the \textbf{eqn}: line. Make your changes, and then press \text{ENTER} or \( \text{C} \) to return to the list of variables.

Specifying an Initial Guess and/or Bounds (Optional)

To find a solution more quickly or to find a particular solution (if multiple solutions exist), you can optionally:

• Enter an initial guess for the unknown variable. The guess must be within the specified bounds.
• Enter lower and upper bounds close to the solution.

For example, \( y_1(x) \) is undefined and you use \( y_1 \).

Initial guess must be within the bounds.
For the bounds, you can also enter variables or expressions that evaluate to appropriate values (\texttt{bound=\{lower,upper\}}) or a valid list variable that contains a two-element list (\texttt{bound=list}). The bounds must be two floating point elements with the first one less than or equal to the second one.

\textbf{Note:} You can also select an initial guess graphically.

**Solving for the Unknown Variable**

After you type an equation in the Numeric Solver and enter values for the known variables, you are ready to solve for the unknown variable.

**Finding the Solution**

With all known variables defined:

1. Move the cursor to the unknown variable.
2. Press \textbf{F2} Solve.
3. A \textbullet{} marks the solution and \texttt{left–rt}. The \textbullet{} disappears when you edit a value, move the cursor to the equation, or leave the solver.
**Note**: To stop (break) a calculation, press [ON]. The unknown variable shows the value being tested when the break occurred.

Using the solution and your entered values, the left and right sides of the equation are evaluated separately. $\text{left–rt}$ shows the difference, which indicates the solution’s accuracy. The smaller the value, the more accurate the solution. If the solution is precise, $\text{left–rt}=0$.

<table>
<thead>
<tr>
<th>If you:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Want to solve for other values</td>
<td>Edit the equation or variable values.</td>
</tr>
<tr>
<td>Want to find a different solution for an equation with multiple solutions</td>
<td>Enter an initial guess and/or a new set of bounds close to the other solution.</td>
</tr>
<tr>
<td>See the message:</td>
<td>Press [ESC]. The unknown variable shows the value being tested when the error occurred.</td>
</tr>
<tr>
<td><img src="image" alt="ERROR" /></td>
<td>• The $\text{left–rt}$ value may be small enough for you to accept the result.</td>
</tr>
<tr>
<td><img src="image" alt="ERROR" /></td>
<td>• If not, enter a different set of bounds.</td>
</tr>
</tbody>
</table>

**Note**: An iterative process is used to solve an equation. If the iterative process cannot converge on a solution, this error occurs.

**Graphing the Solution**

You can graph an equation’s solutions any time after defining the known variables, either before or after you solve for the unknown variable. By graphing the solutions, you can
see how many solutions exist and use the cursor to select an accurate initial guess and bounds.

**Displaying the Graph**

In the Numeric Solver, leave the cursor on the unknown variable. Press F3 and select:

1: Graph View
   – or –
3: ZoomStd
   – or –
4: ZoomFit

The graph is shown in a split screen, where:

- The unknown variable is plotted on the x axis.
- left−rt is plotted on the y axis.

Solutions for the equation exist at left−rt=0, where the graph crosses the x axis.

**Note:** For more information, refer to the *Split Screens* module.
You can explore the graph by using the free-moving cursor, tracing, zooming, etc., as described in *Basic Function Graphing*.

**How the Graph Affects Various Settings**

When you use the Numeric Solver to display a graph:

- The following modes are changed automatically to these settings:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph</td>
<td>FUNCTION</td>
</tr>
<tr>
<td></td>
<td>Any functions selected in the Y= Editor will not be graphed.</td>
</tr>
<tr>
<td>Split Screen</td>
<td>LEFT-RIGHT</td>
</tr>
<tr>
<td>Number of Graphs</td>
<td>1</td>
</tr>
</tbody>
</table>

**Note:** If you were previously using different mode settings, you will need to reselect those settings manually.

- All stat plots are deselected.
- After you leave the Numeric Solver, the Graph screen may continue to display the equation’s solution, ignoring any selected Y= functions. If so, display the Y= Editor and then return to the Graph screen. Also, the graph is reset when you change the Graph mode or use **ClrGraph** from the Home screen (\([F4]\) 5) or a program.
Selecting a New Initial Guess from the Graph

To use the graph cursor to select an initial guess:

1. Move the cursor (either free-moving or trace) to the point that you want to use as the new guess.

2. Use $\text{2nd } [\leftrightarrow]$ to make the Numeric Solver screen active.

3. Make sure the cursor is on the unknown variable, and press $F_4$.

   **Note:** Cursor coordinate $x_c$ is the unknown variable value, and $y_c$ is the left–rt value.

4. Press $F_2$ to re-solve the equation.

Returning to a Full Screen

From the split screen:

- To display the Numeric Solver full screen, use $\text{2nd } [\leftrightarrow]$ to make the solver screen active, press $F_3$, and then select **2:Clear Graph View**.
  
  – or –

- To display the Home screen, press $\text{2nd } [\text{QUIT}]$ twice.
Clearing Variables Before Leaving the Numeric Solver

When you solve an equation, its variables still exist after you leave the Numeric Solver. If the equation contains single-character variables, their values may inadvertently affect later symbolic calculations. Before leaving the Numeric Solver, you may want to:

1. Press:
   
   ![2nd F6](image)

   to clear all single-character variables in the current folder.

2. Press **ENTER** to confirm the action.

   The screen returns to the solver’s `eqn:` line.

**Note:** Any time you want to clear single-character variables listed in the solver, use:

![2nd F6](image).
Number Bases

Entering and Converting Number Bases

Regardless of the Base mode, you must always use the appropriate prefix when entering a binary or hexadecimal number.

**Entering a Binary or Hexadecimal Number**

To enter a binary number, use the form:

```
0b binaryNumber  (for example: 0b11100110)
```

- Binary number with up to 32 digits
- Zero, not the letter O, and the letter b

To enter a hexadecimal number, use the form:

```
0h hexadecimalNumber  (for example: 0h89F2C)
```

- Hexadecimal number with up to 8 digits
- Zero, not the letter O, and the letter h

**Note:** You can type the b or h in the prefix, as well as hex characters A – F, in uppercase or lowercase.

If you enter a number without the 0b or 0h prefix, such as 11, it is always treated as a decimal number. If you omit the 0h prefix on a hexadecimal number containing A – F, all or part of the entry is treated as a variable.
Converting between Number Bases

Use the ▶ conversion operator

\[ \text{integerExpression ▶ Bin} \quad \text{For ▶, press [2nd ▶]. Also, you can} \]
\[ \text{integerExpression ▶ Dec} \quad \text{select base conversions from the} \]
\[ \text{integerExpression ▶ Hex} \quad \text{MATH/Base menu.} \]

For example, to convert 256 from decimal to binary:

\[ 256 ▶ \text{Bin} \]

**Note:** If your entry is not an integer, a Domain error is displayed.

To convert 101110 from binary to hexadecimal:

\[ \text{0b101110 ▶ Hex} \]

**Alternate Method for Conversions**

Instead of using ▶, you can:

1. Use [MODE] to set the Base mode to the base that you want to convert to.

   If Base mode = BIN:

\[ 256 \quad \text{0b1000} \]

\[ 256 \quad \text{MAIN RAD AUTO FUNC} \]

Results use the 0b or 0h prefix to identify.
Performing Math Operations with Hex or Bin Numbers

For any operation that uses an integer number, you can enter a hexadecimal or binary number. Results are displayed according to the Base mode. However, results are restricted to certain size limits when Base = HEX or BIN.

Setting the Base Mode for Displayed Results

1. Press [MODE] [F2] to display Page 2 of the MODE screen.
2. Scroll to the Base mode, press [1], and select the applicable setting.
3. Press [ENTER] to close the MODE screen.
The **Base** mode controls the displayed format of integer results only.

**Note:** The Base mode affects output only. You must always use the **0h** or **0b** prefix to enter a hex or binary number.

Fractional and floating-point results are always shown in decimal form.

**Dividing When Base = HEX or BIN**

When Base=HEX or BIN, a division result is displayed in hexadecimal or binary form only if the result is an integer.

To ensure that division always produces an integer, use `intDiv( )` instead of \( \div \).

If Base mode = HEX:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 0h101101 - 0b101 )</td>
<td>( 0h28 )</td>
</tr>
<tr>
<td>( 254 + 1 )</td>
<td>( 0hFF )</td>
</tr>
<tr>
<td>( 0h5A2C \cdot 6 )</td>
<td>( 0h21D00 )</td>
</tr>
<tr>
<td>( 0hA8F + 0b1001101101 )</td>
<td>( 0hCFC )</td>
</tr>
<tr>
<td>( 0hC45A + 0h6FD2 )</td>
<td>( 0h1342C )</td>
</tr>
<tr>
<td>( 0hc45a+0h6fd2 )</td>
<td>( 0h1342C )</td>
</tr>
</tbody>
</table>

0h prefix in result identifies the base.

If Base mode = HEX:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 0hFF ) ( \div 0h2 )</td>
<td>( 625 )</td>
</tr>
<tr>
<td>( 0hFF ) ( \div 0h2 )</td>
<td>( 127.5 )</td>
</tr>
<tr>
<td><code>intDiv(0hFF,0h2)</code></td>
<td>( 0h7F )</td>
</tr>
<tr>
<td><code>intDiv(0hFF,0h2)</code></td>
<td>( 0h7F )</td>
</tr>
</tbody>
</table>

Press \( \textbf{\text{ABC ENTER}} \) to display the result in APPROXIMATE form.
Size Limitations When Base = HEX or BIN

When Base=HEX or BIN, an integer result is stored internally as a signed, 32-bit binary number, which uses the range (shown in hexadecimal and decimal):

\[
\begin{array}{ccc}
0hFFFFFFF & \quad & 0h1 \\
-1 & \quad & 1 \\
0h80000000 & \quad & 0h0 \\
-2,147,483,648 & \quad & 0 \\
0h7FFFFFFF & \quad & 2,147,483,647 \\
\end{array}
\]

If a result’s magnitude is too large to be stored in a signed, 32-bit binary form, a symmetric modulo operation brings the result into the range. Any number greater than 0h7FFFFFFF is affected. For example, 0h80000000 through 0h7FFFFFFF become negative numbers.

Comparing or Manipulating Bits

The following operators and functions let you compare or manipulate bits in a binary number. You can enter an integer in any number base. Your entries are converted to binary automatically for the bitwise operation, and results are displayed according to the Base mode.

<table>
<thead>
<tr>
<th>Boolean Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operator with syntax</strong></td>
</tr>
<tr>
<td><strong>not</strong> integer</td>
</tr>
<tr>
<td>[−] integer</td>
</tr>
<tr>
<td>Operator with syntax</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>integer1 and integer2</td>
</tr>
<tr>
<td>integer1 or integer2</td>
</tr>
<tr>
<td>integer1 xor integer2</td>
</tr>
</tbody>
</table>

**Note**: You can select these operators from the MATH/Base menu. For an example using each operator, refer to the *Technical Reference* module.

Suppose you enter:

```
0h7AC36 and 0h3D5F
```

Internally, the hexadecimal integers are converted to a signed, 32-bit binary number.

Then corresponding bits are compared.

---

If Base mode = HEX:

```
0h7AC36 and 0h3D5F
0h2C16
```

---

If Base mode = BIN:

```
0b101100000010110
0b101100000010110
```
0h7AC36 = 0b000000000000001111010110000110110
and and
0h3D5F 0b0000000000000000011110101011111
0b0000000000000000000000010110000010110 = 0h2C16
Leading zeros are not shown in the result.

Note: If you enter an integer that is too large to be stored in a signed, 32-bit binary form, a symmetric modulo operation brings the value into the range.

The result is displayed according to the Base mode.

### Rotating and Shifting Bits

<table>
<thead>
<tr>
<th>Function with syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotate(&lt;integer&gt;)</td>
<td>If #ofRotations is:</td>
</tr>
<tr>
<td>– or –</td>
<td>• omitted — bits rotate once to the right (default is -1).</td>
</tr>
<tr>
<td>rotate(&lt;integer&gt;,&lt;#ofRotations&gt;)</td>
<td>• negative — bits rotate the specified number of times to the right.</td>
</tr>
<tr>
<td></td>
<td>• positive — bits rotate the specified number of times to the left.</td>
</tr>
</tbody>
</table>

In a right rotation, the rightmost bit rotates to the leftmost bit; vice versa for a left rotation.
Function with syntax | Description
--- | ---
`shift(integer)` | If `#ofShifts` is:
- omitted — bits shift once to the right (default is -1).
- negative — bits shift the specified number of times to the right.
- positive — bits shift the specified number of times to the left.

In a right shift, the rightmost bit is dropped and 0 or 1 is inserted to match the leftmost bit. In a left shift, the leftmost bit is dropped and 0 is inserted as the rightmost bit.

Suppose you enter:

```
shift(0h7AC36)
```

Internally, the hexadecimal integer is converted to a signed, 32-bit binary number.

Then the shift is applied to the binary number.
Each bit shifts to the right.

7AC36 = 0b00000000000000001111010110000110110

Inserts 0 if leftmost bit is 0, or 1 if leftmost bit is 1

Dropped

b000000000000000000111101011000011011 = 0h3D61B

Leading zeros are not shown in the result.

The result is displayed according to the Base mode.

**Note:** If you enter an integer that is too large to be stored in a signed, 32-bit binary form, a symmetric modulo operation brings the value into the range.
Memory and Variable Management

Checking and Resetting Memory

The MEMORY screen shows the amount of memory (in bytes) used by all variables in each data type, regardless of whether the variables are stored in RAM or the user data archive. You can also use this screen to reset the memory.

Displaying the MEMORY Screen

Press 2nd MEM. (The numbers on your MEMORY screen may vary from those shown.)

Prgm/Asn: Includes programs written for the TI-89 Titanium as well as any assembly-language programs you have loaded.
History: Size of history pairs saved in the Home screen’s history area.
FlashApp: Size of Flash applications.
RAM free: Free space in RAM.
Flash ROM free: Free space in Flash ROM.

Note: To display the size of individual variables and determine if they are in the user data archive, use the VAR-LINK screen.

To close the screen, press ENTER. To reset the memory, use the following procedure.
Resetting the Memory

From the MEMORY screen:

1. Press \[ F_1 \].
2. Select the applicable item.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM 1:All RAM</td>
<td>Resetting RAM erases all data and programs from RAM.</td>
</tr>
<tr>
<td></td>
<td><strong>2:Default:</strong> Resets all system variables and modes to their original factory settings. This does not affect any user-defined variables, functions, or folders.</td>
</tr>
<tr>
<td>Flash ROM 1:Archive</td>
<td>Resetting Archive erases all data and programs from Flash ROM.</td>
</tr>
<tr>
<td></td>
<td><strong>2:Flash Apps:</strong> Resetting Flash Apps erases all Flash applications from Flash ROM.</td>
</tr>
<tr>
<td></td>
<td><strong>3:Both:</strong> Resetting both erases all data, programs, and Flash applications from Flash ROM.</td>
</tr>
<tr>
<td>All Memory</td>
<td>Resetting will delete all data, programs, and Flash applications from RAM and Flash ROM.</td>
</tr>
</tbody>
</table>

**Important:** To delete individual (instead of all) variables, use VAR-LINK.

3. When prompted for confirmation, press \[ \text{ENTER} \].
   The TI-89 Titanium displays a message when the reset is complete.

   **Note:** To cancel the reset, press \[ \text{ESC} \] instead of \[ \text{ENTER} \].

4. Press \[ \text{ENTER} \] to acknowledge the message.
Displaying the VAR-LINK Screen

The **VAR-LINK** screen lists the variables and folders that are currently defined. After displaying the screen, you can manipulate the variables and/or folders.

Displaying the VAR-LINK Screen

Press [2nd] **[VAR-LINK]**. By default, the **VAR-LINK** screen lists all user-defined variables in all folders and with all data types.

1. **Folder names (alphabetically listed)**
2. **Shows installed Flash applications**
3. **Size in bytes**
4. **Data type**
5. **Variable names (alphabetically listed)**

<table>
<thead>
<tr>
<th>This...</th>
<th>Indicates this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶</td>
<td>Collapsed folder view (to right of folder name).</td>
</tr>
<tr>
<td>This...</td>
<td>Indicates this...</td>
</tr>
<tr>
<td>---------</td>
<td>------------------</td>
</tr>
<tr>
<td>▼</td>
<td>Expanded folder view (to right of folder name).</td>
</tr>
<tr>
<td>▼</td>
<td>You can scroll for more variables and/or folders (in bottom left corner of screen).</td>
</tr>
<tr>
<td>✓</td>
<td>If selected with [F4].</td>
</tr>
<tr>
<td>❗</td>
<td>Locked</td>
</tr>
<tr>
<td>❗</td>
<td>Archived</td>
</tr>
</tbody>
</table>

To scroll through the list:

- Press ◄ or ►. (Use 2nd ◄ or 2nd ► to scroll one page at a time.)
  – or –
- Type a letter. If there are any variable names that start with that letter, the cursor moves to highlight the first of those variable names.

**Note:** Type a letter repeatedly to cycle through the names that start with that letter.
Variable Types as Listed on VAR-LINK

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM</td>
<td>Assembly-language program</td>
</tr>
<tr>
<td>DATA</td>
<td>Data</td>
</tr>
<tr>
<td>EXPR</td>
<td>Expression (includes numeric values)</td>
</tr>
<tr>
<td>FUNC</td>
<td>Function</td>
</tr>
<tr>
<td>GDB</td>
<td>Graph database</td>
</tr>
<tr>
<td>LIST</td>
<td>List</td>
</tr>
<tr>
<td>MAT</td>
<td>Matrix</td>
</tr>
<tr>
<td>PIC</td>
<td>Picture of a graph</td>
</tr>
<tr>
<td>PRGM</td>
<td>Program</td>
</tr>
<tr>
<td>STR</td>
<td>String</td>
</tr>
<tr>
<td>TEXT</td>
<td>Text Editor session</td>
</tr>
</tbody>
</table>

Types not listed above are miscellaneous data types used by software applications.

Closing the VAR-LINK Screen

To close the VAR-LINK screen and return to the current application, use [ENTER] or [ESC] as described below.

Press: | To:
---: | ---:
[ENTER] | Paste the highlighted variable or folder name to the cursor location in the current application.
Displaying Information about Variables on the Home Screen

From the Home screen, you can display information about variables without opening the VAR-LINK screen.

• To determine if a variable with a given name exists in the system table, Enter the IsVar() function on the Home screen.

  \[
  \text{IsVar} \left( var\_name \right)
  \]

  \text{IsVar} \] is a function, which requires you to enclose the variable name in parentheses.

• To determine if a variable is archived, use the IsArchiv() function.

  \[
  \text{IsArchiv} \left( var\_name \right)
  \]

• To determine if a variable is locked, use the IsLocked() function.

  \[
  \text{IsLocked} \left( var\_name \right)
  \]

Manipulating Variables and Folders with VAR-LINK

On the VAR-LINK screen, you can show the contents of a variable. You can also select one or more listed items and manipulate them by using the operations in this section.

Press: To:

[ESC] Return to the current application without pasting the highlighted name.
**Showing the Contents of a Variable**

You can show all variable types except **ASM, DATA, GDB, and variables created by Flash Apps**. For example, you must open a **DATA** variable in the Data/Matrix Editor.

1. On **VAR-LINK**, move the cursor to highlight the variable.

2. Press: $2nd$ $[F6]$ 

   If you highlight a folder, the screen shows the number of variables in that folder.

3. To return to **VAR-LINK**, press any key.

**Note:** You cannot edit the contents from this screen.

**Selecting Items from the List**

For other operations, select one or more variables and/or folders.

<table>
<thead>
<tr>
<th>To select:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A single variable or folder</td>
<td>Move the cursor to highlight the item, then press $[F4]$</td>
</tr>
<tr>
<td>A group of variables or folders</td>
<td>Highlight each item and press $[F4]$. A $\checkmark$ is displayed to the left of each selected item. (If you select a folder, all variables in that folder are selected.) Use $[F4]$ to select or deselect an item.</td>
</tr>
</tbody>
</table>
Folders and Variables

Folders give you a convenient way to manage variables by organizing them into related groups.

The TI-89 Titanium has one built-in folder named MAIN. Unless you create other folders and designate a user-created folder as the current folder, all variables are stored in the MAIN folder by default. A system variable or a variable with a reserved name can be stored in the MAIN folder only.

Example of variables that can be stored in MAIN only

- Window variables
  - \( \text{xmin}, \text{xmax} \), etc.
- Table setup variables
  - \( \text{TblStart}, \Delta \text{Tbl} \), etc.
- \( Y= \) Editor functions
  - \( y1(x) \), etc.
By creating additional folders, you can store independent sets of user-defined variables (including user-defined functions). For example, you can create separate folders for different TI-89 Titanium applications (Math, Text Editor, etc.) or classes. You can store a user-defined variable in any existing folder.

The user-defined variables in one folder are independent of the variables in any other folder. Therefore, folders can store separate sets of variables with the same names but different values.

You cannot create a folder within another folder.
The system variables in the **MAIN** folder are always directly accessible, regardless of the current folder.

**Note:** User-defined variables are stored in the “current folder” unless you specify otherwise.

**Creating a Folder from the VAR-LINK Screen**


2. Press `F1 Manage` and select `5:Create Folder`.  

3. Type a unique folder name up to eight characters, and press `[ENTER]` twice.

After you create a new folder from **VAR-LINK**, that folder is not automatically set as the current folder.

**Creating a Folder from the Home Screen**

Enter the **NewFold** command on the Home screen.

```
NewFold folderName
```

Folder name to create. This new folder is set automatically as the current folder.
Setting the Current Folder from the Home Screen

Enter the `setFold` function on the Home screen.

```
setFold(folderName)
```

- `setFold` is a function, which requires you to enclose the folder name in parentheses.

When you execute `setFold`, it returns the name of the folder that was previously set as the current folder.

Setting the Current Folder from the MODE Dialog Box


2. Highlight the Current Folder setting.

3. Press `[#]` to display a menu of existing folders.

   **Note:** To cancel the menu or exit the dialog box without saving any changes, press `[ESC]`.

4. Select the applicable folder. Either:
   - Highlight the folder name and press `[ENTER]`.
     - or -
   - Press the corresponding number or letter for that folder.

5. Press `[ENTER]` to save your changes and close the dialog box.
Renaming Variables or Folders

Remember, if you use [F4] to select a folder, the variables in that folder are selected automatically. As necessary, use [F4] to deselect individual variables.

1. On VAR-LINK, select the variables and/or folders.

3. Type a unique name, and press [ENTER] twice.
   If you selected multiple items, you are prompted to enter a new name for each one.

Using Variables in Different Folders

You can access a user-defined variable or function that is not in the current folder. Specify the complete pathname instead of only the variable name.

A pathname has the form:

folderName \ variableName
– or –
folderName \ functionName
For example:

<table>
<thead>
<tr>
<th>If Current Folder = MAIN</th>
<th>Folders and Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 \to a</td>
<td>MAIN</td>
</tr>
<tr>
<td>( x^3 + x^2 + x + f(x) )</td>
<td>a=1</td>
</tr>
<tr>
<td>42 \to \text{math}\backslash a</td>
<td>f(x)=x^3+x^2+x</td>
</tr>
<tr>
<td>3 \cdot x^2 + 4 \cdot x + 25 \to \text{math}\backslash f(x)</td>
<td>MATH</td>
</tr>
<tr>
<td>( 3 \cdot x^2 + 4 \cdot x + 25 + \text{math}\backslash f(x) )</td>
<td>a=42</td>
</tr>
<tr>
<td></td>
<td>f(x)=3x^2+4x+25</td>
</tr>
</tbody>
</table>

To see a list of existing folders and variables, press \([2nd] \text{[VAR-LINK]}\). On the \text{VAR-LINK} screen, you can highlight a variable and press \([\text{ENTER}]\) to paste that variable name to the open application's entry line. If you paste a variable name that is not in the current folder, the pathname \((\text{folderName}\backslash\text{variableName})\) is pasted.

**Listing Only a Specified Folder and/or Variable Type, or Flash application**

If you have a lot of variables, folders, or Flash applications, it may be difficult to locate a particular variable. By changing \text{VAR-LINK}'s view, you can specify the information you want to see.
From the **VAR-LINK** screen:

1. Press **F2 View**.

2. Highlight the setting you want to change, and press **§**. This displays a menu of valid choices. (To cancel a menu, press **ESC**.)

   **View** — Allows you to choose variables, Flash applications, or system variables to view.

   **Note:** To list system variables (window variables, etc.), select **3:System**.

   **Folder** — Always lists **1:All** and **2:main**, but lists other folders only if you have created them.

   **Var Type** — Lists the valid variable types.

   ↓ — indicates that you can scroll for additional variable types.

3. Select the new setting.

4. When you are back on the **VAR-LINK VIEW** screen, press **ENTER**.

The **VAR-LINK** screen is updated to show only the specified folder, variable type, or Flash application.
Copying or Moving Variables from One Folder to Another

You must have at least one folder other than MAIN. You cannot use VAR-LINK to copy variables within the same folder.

1. On VAR-LINK, select the variables.
2. Press F1 Manage and select 2:Copy or 4:Move.
3. Select the destination folder.
4. Press ENTER. The copied or moved variables retain their original names.
   
   Note: To copy a variable to a different name in the same folder, use STO (such as a1→a2) or the CopyVar command from the Home screen.

Locking or Unlocking Variables Folders, or Flash Applications

When a variable is locked, you cannot delete, rename, or store to it. However, you can copy, move, or display its contents. When a folder is locked, you can manipulate the variables in the folder (assuming the variables are not locked), but you cannot delete the folder. When a Flash application is locked, you cannot delete it.

1. On VAR-LINK, select the variables, folders, or Flash application.
2. Press \textbf{F1 Manage} and select \textbf{6:Lock} or \textbf{7:UnLock}.

\(\text{\textbullet} \) indicates a locked variable or folder in RAM.

\(\text{\textbullet} \) indicates an archived variable, which is locked automatically.

### Deleting a Folder from the VAR-LINK Screen

When you delete a folder from the \textbf{VAR-LINK} screen, all of the variables in that folder are also deleted. You cannot delete the \textbf{MAIN} folder.

1. Press \textbf{2nd [VAR-LINK]}.

2. Press \textbf{F4} to select the folder(s) to delete. (The folder's variables become selected automatically.)

3. Press \textbf{F1 1:Delete} or \textbf{\(\leftarrow\)}.

4. Press \textbf{ENTER} to confirm the deletion of the folder and all its variables.
Deleting a Variable or a Folder from the Home Screen

Before deleting a folder from the Home screen, you must first delete all the variables stored in that folder.

- To delete a variable, enter the **DelVar** command on the calculator Home screen.

  \[
  \text{DelVar } var1 [, var2] [, var3] ... 
  \]

- To delete all variables of a specific type, enter the **DelType** command on the calculator Home screen.

  \[
  \text{DelType } var\_type \text{ where } var\_type \text{ is the variable type.} \]

  **Note:** The **DelType** command deletes all variables of the specified type in all folders.

- To delete an empty folder, enter the **DelFold** command on the calculator Home screen.

  \[
  \text{DelFold } folder1 [, folder2] [, folder3] ... 
  \]

  **Note:** You cannot delete the **MAIN** folder.

Pasting a Variable Name to an Application

Suppose you are typing an expression on the Home screen and can’t remember which variable to use. You can display the **VAR-LINK** screen, select a variable from the list, and paste that variable name directly onto the Home screen’s entry line.

Which Applications Can You Use?

From the following applications, you can paste a variable name to the current cursor location.
• Home screen, Y= Editor, Table Editor, or Data/Matrix Editor — The cursor must be on the entry line.

• Text Editor, Window Editor, Numeric Solver, or Program Editor — The cursor can be anywhere on the screen.

You can also paste a variable name to the current cursor location in many Flash applications.

**Procedure**

Starting from an application listed above:

1. Position the cursor where you want to insert the variable name.

```
\text{sin(}\)
```

2. Press \text{2nd} \text{[VAR-LINK]}.

3. Highlight the applicable variable.

\text{Note:} You can also highlight and paste folder names.

```
\text{sin(a1)}
```

4. Press \text{ENTER} to paste the variable name.

\text{Note:} This pastes the variable’s name, not its contents. Use \text{2nd} \text{[RCL]}, instead of \text{2nd} \text{[VAR-LINK]}, to recall a variable’s contents.

5. Finish typing the expression.

```
\text{sin(a1)}\]
```
If you paste a variable name that is not in the current folder, the variable’s pathname is pasted.

Assuming that CLASS is *not* the current folder, this is pasted if you highlight the a2 variable in CLASS.

**Archiving and Unarchiving a Variable**

To archive or unarchive one or more variables interactively, use the VAR-LINK screen. You can also perform these operations from the Home screen or a program.

**Why Would You Want to Archive a Variable?**

The user data archive lets you:

- Store data, programs, or any other variables to a safe location where they cannot be edited or deleted inadvertently.
- Create additional free RAM by archiving variables. For example:
  - You can archive variables that you need to access but do not need to edit or change, or variables that you are not using currently but need to retain for future use.

  **Note:** You cannot archive variables with reserved names or system variables.
  - If you acquire additional programs for your TI-89 Titanium, particularly if they are large, you may need to create additional free RAM before you can install those programs.
Additional free RAM can improve performance times for certain types of calculations.

**From the VAR-LINK Screen**

To archive or unarchive:

1. Press `[2nd] [VAR-LINK]` to display the **VAR-LINK** screen.

2. Select one or more variables, which can be in different folders. (You can select an entire folder by selecting the folder name.)
   
   **Note:** To select a single variable, highlight it. To select multiple variables, highlight each variable and press `[F4] ✓`.

3. Press `[F1]` and select either:
   
   **8:Archive Variable**
   
   – or –
   
   **9:Unarchive Variable**

   If you select **8:Archive Variable**, the variables are moved to the user data archive.

   \[ \times = \text{archived variables} \]

   You can access an archived variable just as you would any locked variable. For all purposes, an archived variable is still in its original folder; it is simply stored in the user data archive instead of RAM.

   **Note:** An archived variable is locked automatically. You can access the variable, but you cannot edit or delete it.
From the Home Screen or a Program

Use the Archive and Unarchiv commands:

Archive variable1, variable2, ...

Unarchiv variable1, variable2, ...

If a Garbage Collection Message Is Displayed

If you use the user data archive extensively, you may see a Garbage Collection message. This occurs if you try to archive a variable when there is not enough free archive memory. However, the TI-89 Titanium will attempt to rearrange the archived variables to make additional room.

Responding to the Garbage Collection Message

When you see the message to the right:

• To continue archiving, press ENTER.

– or –

• To cancel, press ESC.

Note: If batteries are low, replace them before performing garbage collection, or archive memory may be lost.

After garbage collection, depending on how much additional space is freed, the variable may or may not be archived. If not, you can unarchive some variables and try again.
Why not Perform Garbage Collection Automatically, without a Message?

The message:

- Lets you know why an archive will take longer than usual. It also alerts you that the archive may fail if there is not enough memory.
- Can alert you when a program is caught in a loop that repetitively fills the user data archive. Cancel the archive and investigate the reason.

Why Is Garbage Collection Necessary?

The user data archive is divided into sectors. When you first begin archiving, variables are stored consecutively in sector 1. This continues to the end of the sector. If there is not enough space left in the sector, the next variable is stored at the beginning of the next sector. Typically, this leaves an empty block at the end of the previous sector.

Each variable that you archive is stored in the first empty block large enough to hold it.

**Note:** An archived variable is stored in a continuous block within a single sector; it cannot cross a sector boundary.
This process continues to the end of the last sector. Depending on the size of individual variables, the empty blocks may account for a significant amount of space.

**Note:** Garbage collection occurs when the variable you are archiving is larger than any empty block.

**How Unarchiving a Variable Affects the Process**

When you unarchive a variable, it is copied to RAM but it is not actually deleted from user data archive memory.
Unarchived variables are “marked for deletion,” meaning they will be deleted during the next garbage collection.

**If the MEMORY Screen Shows Enough Free Space**

Even if the MEMORY screen shows enough free space to archive a variable, you may still get a Garbage Collection message.

This TI-89 Titanium memory screen shows free space that will be available after all “marked for deletion” variables are deleted.

When you unarchive a variable, the Flash ROM free amount increases immediately, but the space is not actually available until after the next garbage collection.
The Garbage Collection Process

The garbage collection process:

• Deletes unarchived variables from the user data archive.
• Rearranges the remaining variables into consecutive blocks.

Memory Error When Accessing an Archived Variable

An archived variable is treated the same as a locked variable. You can access the variable, but you cannot edit or delete it. In some cases, however, you may get a Memory Error when you try to access an archived variable.

What Causes the Memory Error?

The Memory Error message is displayed if there is not enough free RAM to access the archived variable. This may cause you to ask, “If the variable is in the user data archive, why does it matter how much RAM is available?” The answer is that the following operations can be performed only if a variable is in RAM.
• Opening a text variable in the Text Editor.
• Opening a data variable, list, or matrix in the Data/Matrix Editor.
• Opening a program or function in the Program Editor.
• Running a program or referring to a function.

Note: A temporary copy lets you open or execute an archived variable. However, you cannot save any changes to the variable.

So that you do not have to unarchive variables unnecessarily, the TI-89 Titanium performs a “behind-the scenes” copy. For example, if you run a program that is in the user data archive, the TI-89 Titanium:

1. Copies the program to RAM.
2. Runs the program.
3. Deletes the copy from RAM when the program is finished.

The error message is displayed if there is not enough free RAM for the temporary copy.

Note: Except for programs and functions, referring to an archived variable does not copy it. If variable ab is archived, it is not copied if you perform 6*ab.

Correcting the Error

To free up enough RAM to access the variable:

1. Use the VAR-LINK screen (2nd [VAR-LINK]) to determine the size of the archived variable that you want to access.
2. Use the MEMORY screen (2nd [MEM]) to check the RAM free size.
3. Free up the needed amount of memory by:
   • Deleting unnecessary variables from RAM.
   • Archiving large variables or programs (moving them from RAM to the user data archive).

**Note**: Typically, the RAM free size must be larger than the archived variable.
Connectivity

Connecting Two Units

The TI-89 Titanium comes with a cable that lets you connect two units. Once connected, you can transmit information between two units. A USB unit-to-unit cable is included with the TI-89 Titanium; use the calculator’s USB port with this cable.

**Note:** The TI-89 Titanium features both a USB port and an I/O port, so you can connect TI graphing calculators with either type of link port. However, using the I/O port requires the I/O unit-to-unit cable (sold separately) or the USB Silver Edition cable (also sold separately), which is used to connect to a computer.

**Connecting before Sending or Receiving**

Using firm pressure, insert one end of the cable into the link port of each unit. Either unit can send or receive, depending on how you set them up from the VAR-LINK screen.

You can link a TI-89 Titanium or Voyage™ 200 to another TI-89 Titanium, Voyage™ 200, TI-89, or TI-92 Plus.
Two TI-89 Titanium calculators linked together

Position so that the USB symbols face each other; then insert the connector.
A TI-89 Titanium and a Voyage™ 200 linked together
Transmitting Variables, Flash Applications, and Folders

Transmitting variables is a convenient way to share any variable listed on the VAR-LINK screen — functions, programs, etc. You can also transmit Flash applications (Apps) and folders.
Setting Up the Units

Flash applications will transfer only between certain units. For example, you can transfer an App from a TI-89 Titanium to another TI-89 Titanium, or from a TI-89 Titanium to a TI-89.

1. Connect two graphing calculators using the appropriate cable.

2. On the *sending* unit, press [2nd][VAR-LINK] to display the **VAR-LINK** screen.

3. On the *sending* unit, select the variables, folders, or Flash applications you want to send.

   - To select a single variable, Flash application, or folder, move the cursor to highlight it and press [F4] to place a checkmark (✓) beside it.

   - If on the default **VAR-LINK** screen, this selects the folder and its contents. Collapsed folders become expanded when selected.
- If selecting a Flash App (from the F7 tab), this selects the App folder and its contents. A checkmark appears beside the folder, but not beside the contents. Collapsed Flash App folders do not automatically become expanded.

- To select multiple variables, Flash applications, or folders, highlight each one and press [F4] to place a checkmark (√) beside it. Use [F4] again to deselect any that you do not want to transmit.

- To select all variables, Flash applications, or folders use [F5] All 1:Select All.

4. On the receiving unit, press [2nd] [VAR-LINK] to display the VAR-LINK screen. (The sending unit remains on the VAR-LINK screen.)
5. On both the receiving and the sending unit, press \[F3\] Link to display the menu options.

6. On the receiving unit, select 2:Receive.
   The message **VAR-LINK: WAITING TO RECEIVE** and the BUSY indicator are displayed in the status line of the receiving unit.

7. On the sending unit, select 1:Send
   This starts the transmission.

During transmission, a progress bar is displayed in the status line of the receiving unit. When transmission is complete, the **VAR-LINK** screen is updated on the receiving unit.

**Note:** Before transferring a purchased App, the receiving unit must have the appropriate certificate, if required. A certificate is a file that is generated by TI. Free and concept Apps do not require a certificate.

**Rules for Transmitting Variables, Flash Applications, or Folders**

Unlocked and unarchived variables that have the same name on both the sending and receiving units will be overwritten from the sending unit.
Locked variables that have the same name on both the sending and receiving units must be unlocked on the receiving unit before they can be overwritten from the sending unit. If archived variables have the same names on both the sending and receiving units, a message asks you to confirm that you will allow the variables to be overwritten.

<table>
<thead>
<tr>
<th>If you select:</th>
<th>What happens:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unlocked variable</td>
<td>The variable is transmitted to the current folder and it remains unlocked on the receiving unit.</td>
</tr>
<tr>
<td>Locked variable</td>
<td>The variable is transmitted to the current folder and it remains locked on the receiving unit.</td>
</tr>
<tr>
<td>Archived variable</td>
<td>The variable is transmitted to the current folder and it remains archived on the receiving unit.</td>
</tr>
<tr>
<td>Unlocked Flash application</td>
<td>If the receiving unit has the correct certification, the Flash application is transmitted. It remains unlocked on the receiving unit.</td>
</tr>
<tr>
<td>Locked Flash application</td>
<td>If the receiving unit has the correct certification, the Flash application is transmitted. It remains locked on the receiving unit.</td>
</tr>
<tr>
<td>Unlocked Folder</td>
<td>The folder and its selected contents are transmitted. The folder remains unlocked on the receiving unit.</td>
</tr>
<tr>
<td>Locked Folder</td>
<td>The folder and its selected contents are transmitted. The folder becomes unlocked on the receiving unit.</td>
</tr>
</tbody>
</table>
Canceling a Transmission

From either the sending or receiving unit:

1. Press \textbf{ON}.
   
   An error message is displayed.

2. Press \textbf{ESC} or \textbf{ENTER}.

Common Error and Notification Messages

<table>
<thead>
<tr>
<th>Shown on:</th>
<th>Message and Description:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sending unit</td>
<td><img src="image" alt="ERROR" /></td>
</tr>
</tbody>
</table>

This is displayed after several seconds if:

- A cable is not attached to the sending unit’s link port.
  - or -
- A receiving unit is not attached to the other end of the cable.
  - or -
- The receiving unit is not set up to receive.

Press \textbf{ESC} or \textbf{ENTER} to cancel the transmission.

\textbf{Note:} The sending unit may not always display this message. Instead, it may remain \textbf{BUSY} until you cancel the transmission.
The receiving unit does not have the correct certification for the operating system (OS) or Flash application being sent.

The receiving unit has a variable with the same name as the specified variable being sent.

- To overwrite the existing variable, press [ENTER]. (By default, Overwrite = YES.)
- To store the variable to a different name, set Overwrite = NO. In the New Name input box, type a variable name that does not exist in the receiving unit. Then press [ENTER] twice.
- To skip this variable and continue with the next one, set Overwrite = SKIP and press [ENTER].
- To cancel the transmission, press [ESC].

New Name is active only if you change Overwrite to NO.
Deleting Variables, Flash Applications, or Folders

1. Press \[2^\text{nd}\] [VAR-LINK] to display the VAR-LINK screen.

2. Select the variables, folders, or Flash applications to delete.
   
   - To select a single variable, Flash application, or folder, move the cursor to highlight it and press \[F4\] to place a checkmark (✓) beside it.
     
     - If on the default VAR-LINK screen, this selects the folder and its contents. Collapsed folders become expanded when selected.
     
     - If selecting a Flash App (from the F7 tab), this selects the App folder and its contents. A checkmark appears beside the folder, but not beside the contents. Collapsed Flash App folders do not automatically become expanded.

   \textbf{Note:} You cannot delete the Main folder.

   - To select multiple variables, Flash applications, or folders highlight each one and press \[F4\] to place a checkmark (✓) beside it. Use \[F4\] again to deselect any that you do not want to transmit.

   - To select all variables, Flash applications, or folders use \[F5\] \textit{All 1:Select All}. 

The receiving unit does not have enough memory for what is being sent. Press \[\text{ESC}\] or \[\text{ENTER}\] to cancel the transmission.
3. Press \( F_1 \) and choose 1:Delete.
   – or –
   Press \( \leftarrow \). A confirmation message appears.

4. Press \( \text{ENTER} \) to confirm the deletion.

**Where to Get Flash Applications (Apps)**

For up-to-date information about available Flash applications, check the Texas Instruments Web site at education.ti.com.

Many Apps no longer require a certificate. If you try to transfer an App from one unit to another and receive an **Unlicensed OS or Flash application** message, try downloading the App again from the Texas Instruments Web site at education.ti.com.

You can download a Flash application and/or certificate from the Texas Instruments Web site to a computer, and use a TI Connectivity Cable USB to install the application or certificate on your TI-89 Titanium.

For Flash App installation instructions, see education.ti.com/guides.

**Transmitting Variables under Program Control**

You can use a program containing \( \text{GetCalc} \) and \( \text{SendCalc} \) to transmit a variable from one device to another.

\( \text{SendCalc} \) sends a variable to the link port, where a linked device can receive the variable. The linked device must be on the Home screen or must execute \( \text{GetCalc} \) from a program.
You can use optional parameters with the SendCalc or GetCalc command to specify either the USB port or I/O port. (See Appendix A for details.) If you do not include these parameters, the TI-89 Titanium communicates through the USB port.

**The “Chat” Program**

The following program uses **GetCalc** and **SendCalc**. The program sets up two loops that let the linked devices take turns sending and receiving/displaying a variable named **msg**. **InputStr** lets each user enter a message in the **msg** variable.
Notes:

1. Sets up this unit to receive and display the variable msg.
2. Then lets this user enter a message in msg and send it.
3. Loop executed by the unit that receives the first message.
4. Lets this user enter a message in msg and send it.
5. Then sets up this unit to receive and display msg.
6. Loop executed by the unit that sends the first message.
To synchronize GetCalc and SendCalc, the loops are arranged so that the receiving unit executes GetCalc while the sending unit is waiting for the user to enter a message.

**Running the Program**

This procedure assumes that:

- The two devices are linked with the connecting cable.
- The Chat program is loaded on both devices.
  - Use each device’s Program Editor to enter the program.
    – or –
  - Enter the program on one device and then use VAR-LINK to transmit the program variable to the other device.

To run the program on both devices:

1. On the Home screen of each device, enter chat().
2. When each device displays its initial prompt, respond as shown below.

<table>
<thead>
<tr>
<th>On the:</th>
<th>Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device that will send the first message.</td>
<td>1 and press [ENTER].</td>
</tr>
<tr>
<td>Device that will receive the first message.</td>
<td>0 and press [ENTER].</td>
</tr>
</tbody>
</table>

3. Take turns typing a message and pressing [ENTER] to send the variable msg to the other device.
Stopping the Program

Because the Chat program sets up an infinite loop on both devices, press ON (on both devices) to break the program. If you press ESC to acknowledge the error message, the program stops on the Program I/O screen. Press F5 or ESC to return to the Home screen.

Upgrading the Operating System (OS)

You can upgrade the OS on your TI-89 Titanium using your computer. You can also transfer the OS from one unit to another identical model (for example, from a TI-89 Titanium to a TI-89 Titanium or from a Voyage™ 200 to a Voyage™ 200).

Installing OS software resets all device memory to the original factory settings. This means that all user-defined variables (in both RAM and the user data archive), functions, programs, lists, and folders (except the Main folder) will be deleted. It is possible that Flash applications could also be deleted. You should use TI Connect software to back up your data to your computer before installing a new OS on your calculator.

See the important information concerning batteries before performing an OS upgrade.

Important Operating System Download Information

New batteries should be installed before beginning an OS download.

If you are operating your TI-89 Titanium in a language other than English, you should ensure that you have the most current localizer application installed when you upgrade the OS software. If you do not have the most current localizer installed, prompts, error
messages and status information related to new functionality in the OS may not display correctly.

When in OS download mode, the Automatic Power Down™ (APD™) feature does not function. If you leave your device in download mode for an extended time before you actually start the downloading process, your batteries may become depleted. You will then need to replace the depleted batteries with new batteries before downloading.

If you accidentally interrupt the transfer before it is complete, you will need to reinstall the OS. Again, remember to install new batteries before downloading.

**Backing Up Your Unit Before an Operating System Installation**

When you install an OS upgrade, the installation process:

- Deletes all user-defined variables (in both RAM and the user data archive), functions, programs, and folders.
- Could delete all Flash applications.
- Resets all system variables and modes to their original factory settings. This is equivalent to using the **MEMORY** screen to reset all memory.

To retain any existing variables or Flash applications, do the following before installing the upgrade:

- **Important:** Install new batteries.
- Transmit the variables or Flash applications to another device. – or –
• Use a USB cable or TI Connectivity Cable USB and TI Connect™ software (education.ti.com/downloadticonnect) to send the variables and/or Flash applications to a computer.

**Where to Get Operating System Upgrades**

For up-to-date information about available OS upgrades, check the Texas Instruments Web site at education.ti.com/downloadticonnect.

You can download an OS upgrade, Localizer or Flash application from the Texas Instruments Web site to a computer, and use a USB computer cable to install the OS or application on your TI-89 Titanium.

For complete information, refer to the instructions on the web.

**Transferring the Operating System**

OS software will transfer only from a TI-89 Titanium to a TI-89 Titanium, TI-89 to a TI-89, from a Voyage™ 200 to a Voyage™ 200, or from a TI-92 Plus to a TI-92 Plus.

To transfer the Operating System (OS) from unit to unit:

1. Link two like units together, for example, a TI-89 Titanium to a TI-89 Titanium; or a Voyage™ 200 to a Voyage™ 200.

2. On the receiving and the sending unit, press [2nd] [VAR-LINK] to display the VAR-LINK screen.

3. On the receiving and the sending unit, press [F3] Link to display the menu options.
4. On the receiving unit, select **5:Receive OS**.

   A warning message displays. Press [ESC] to halt the process, or press [ENTER] to proceed. Pressing [ENTER], displays **VAR-LINK: WAITING TO RECEIVE** and **BUSY** in the status line of the receiving unit.

5. On the sending unit, select **4:Send OS**.

   A warning message displays. Press [ESC] to halt the process, or press [ENTER] to start the transmission.

**Important:**

- For each receiving unit, remember to back up information as necessary and install new batteries.
- Be sure both the sending and receiving units are in the **VAR-LINK** screen.

During the transfer, the receiving unit shows how the transfer is progressing. When the transfer is complete:

- The sending unit returns to the **VAR-LINK** screen.
- The receiving unit returns to either the Apps desktop or the Home screen. You may need to use [-] (lighten) or [+](darken) to adjust the contrast.

**Do Not Attempt to Cancel an Operating System Transfer**

After the transfer starts, the receiving unit’s existing OS is effectively deleted. If you interrupt the transfer before it is complete, the receiving unit will not operate properly. You will then need to reinstall the OS upgrade.
If You are Upgrading the Operating System on Multiple Units

To perform an OS upgrade on multiple units, download and install the OS into one unit and then transfer the OS upgrade from one unit to another. This method is faster than installing it on each unit via a computer. OS upgrades are released free of charge and you do not need to obtain a certificate before you download or install them.

Error Messages

Most error messages are displayed on the sending unit. Depending on when the error occurs during the transfer process, you may see an error message on the receiving unit.

<table>
<thead>
<tr>
<th>Error Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Error 1" /></td>
<td>The sending and receiving units are not connected properly, or the receiving unit is not set up to receive.</td>
</tr>
<tr>
<td><img src="image2" alt="Error 2" /></td>
<td>The certificate on the receiving unit is not valid for the operating system (OS) or App on the sending unit. You must obtain and install a valid certificate.</td>
</tr>
<tr>
<td><img src="image3" alt="Error 3" /></td>
<td>If the App no longer requires a certificate, you can download it again from the Texas Instruments Web site at education.ti.com and then install the App again on your calculator.</td>
</tr>
<tr>
<td><img src="image4" alt="Error 4" /></td>
<td>An error occurred during the transfer. The current OS in the receiving unit is corrupted. You must reinstall the product software from a computer.</td>
</tr>
</tbody>
</table>
Collecting and Transmitting ID Lists

The VAR-LINK screen F3 6:Send ID List menu option allows collection of electronic ID numbers from individual TI-89 Titanium, TI-89, Voyage™ 200, or TI-92 Plus devices.

ID Lists and Group Certificates

The ID list feature provides a convenient way to collect device IDs for group purchase of commercial applications. After the IDs are collected, transmit them to Texas Instruments so a group certificate can be issued.

A group certificate allows distribution of purchased software to multiple TI-89 Titanium, TI-89, Voyage™ 200, or TI-92 Plus units. The software can be loaded, deleted from, and reloaded to the devices as often as needed for as long as the software remains listed in the group certificate. You may add new ID numbers and/or new commercial applications to a group certificate.

Collecting ID Lists

You can use one device to collect all of the IDs, or use several collection units and then consolidate their ID lists onto one device.
To send an ID number from one device to another, first connect two units by using a USB unit-to-unit cable or I/O unit-to-unit cable.

<table>
<thead>
<tr>
<th>Step</th>
<th>On the:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Collecting unit</td>
<td>Display the Home screen. Press:HOME</td>
</tr>
<tr>
<td></td>
<td>(Receiving unit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b. Press [F3] Link and select 6:Send ID List.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The sending unit adds a copy of its unique ID number to the collection unit’s ID list. The sending unit always retains its own ID number, which cannot be deleted from the device.</td>
</tr>
<tr>
<td>3.</td>
<td>Additional units</td>
<td>Repeat steps 1 and 2 until all the IDs are collected onto one device.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Depending on available memory in the collection device, it is possible to collect over 4,000 IDs.</td>
</tr>
</tbody>
</table>

**Notes:**

- You cannot view the ID list on the sending or collecting units.
• Each time an ID list is successfully sent from one device to another, the ID list is automatically deleted from the sending unit.

• If an ID is collected from a device twice, the duplicate ID is automatically deleted from the list.

**Clearing the ID List**

The ID list remains on the collection device after it is uploaded to the computer. You can then use the collection device to upload the list to other computers.

To clear the ID list from the collection unit:

1. Press 2nd [VAR-LINK] to display the VAR-LINK screen.

2. Press F1 Manage and select A:Clear ID List.

**Compatibility among the TI-89 Titanium, Voyage™ 200, TI-89, and TI-92 Plus**

In general, TI-89 Titanium, TI-89, Voyage™ 200, and TI-92 Plus data and programs are compatible with each other, with a few exceptions.
Most functions of the TI-89 Titanium are compatible with the TI-89, Voyage™ 200, and TI-92 Plus. The TI-89 Titanium and the TI-89 are similar, except that the TI-89 Titanium has more memory (more room for Apps and user archive) and the TI-89 Titanium has a USB port. The Voyage™ 200 is the same as the TI-92 Plus except it has more memory, and thus more room for applications (Apps).

All data is compatible among the TI-89 Titanium, TI-89, Voyage™ 200, and TI-92 Plus, but some programs written for one may not run or may not run the same on the other because of differences in the device’s screen sizes and keyboards and the USB port on the TI-89 Titanium.

Other incompatibilities can occur because of different version the operating system. To download the latest version of the operating system, visit the Texas Instruments Web site at education.ti.com/downloadticconnect.
## Link Transmission Table

<table>
<thead>
<tr>
<th>To</th>
<th>TI-89 Titanium</th>
<th>TI-89</th>
<th>Voyage™ 200</th>
<th>TI-92 Plus</th>
</tr>
</thead>
<tbody>
<tr>
<td>From ↓</td>
<td>Apps Variables</td>
<td>OS</td>
<td>Apps Variables</td>
<td>OS</td>
</tr>
<tr>
<td>TI-89 Titanium</td>
<td>Apps Variables</td>
<td>Variables</td>
<td>Variables</td>
<td>Variables</td>
</tr>
<tr>
<td>TI-89</td>
<td>Apps Variables</td>
<td>OS Apps Variables</td>
<td>Variables</td>
<td>Variables</td>
</tr>
<tr>
<td>Voyage™ 200</td>
<td>Variables Apps Variables</td>
<td>Variables OS Apps Variables</td>
<td>Apps Variables</td>
<td>Variables</td>
</tr>
<tr>
<td>TI-92 Plus</td>
<td>Variables</td>
<td>Variables Apps</td>
<td>OS Apps Variables</td>
<td>Variables</td>
</tr>
</tbody>
</table>
Activities

Analyzing the Pole-Corner Problem

A ten-foot-wide hallway meets a five-foot-wide hallway in the corner of a building. Find the maximum length pole that can be moved around the corner without tilting the pole.

Maximum Length of Pole in Hallway

The maximum length of a pole $c$ is the shortest line segment touching the interior corner and opposite sides of the two hallways as shown in the diagram below.

Use proportional sides and the Pythagorean theorem to find the length $c$ with respect to $w$. Then find the zeros of the first derivative of $c(w)$. The minimum value of $c(w)$ is the maximum length of the pole.
1. Define the expression for side $a$ in terms of $w$ and store it in $a(w)$.

**Note:** When you want to define a function, use multiple character names as you build the definition.

2. Define the expression for side $b$ in terms of $w$ and store it in $b(w)$.

3. Define the expression for side $c$ in terms of $w$ and store it in $c(w)$.

Enter: Define $c(w) = \sqrt{(a(w)^2 + b(w)^2)}$
4. Use the zeros( ) function to compute the zeros of the first derivative of $c(w)$ to find the minimum value of $c(w)$.

**Note:** The maximum length of the pole is the minimum value of $c(w)$.

5. Compute the exact maximum length of the pole.

Enter: $c\left(\frac{2}{\sqrt{3}}\right)$

6. Compute the approximate maximum length of the pole.

Result: Approximately 20.8097 feet.

**Note:** Use the auto-paste feature to copy the result from step 4 to the entry line inside the parentheses of $c(\ )$ and press ENTER.

---

### Deriving the Quadratic Formula

This activity shows you how to derive the quadratic formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$
Performing Computations to Derive the Quadratic Formula

Perform the following steps to derive the quadratic formula by completing the square of the generalized quadratic equation.

1. Clear all one-character variables in the current folder.

   \[ \text{2nd}[F6] \]

   Choose \textbf{1:Clear a-z} and press \textbf{ENTER} to confirm.

2. On the Home screen, enter the generalized quadratic equation:

   \[ ax^2 + bx + c = 0. \]

3. Subtract \( c \) from both sides of the equation.

   \[ \text{2nd}[\text{ANS}] - \alpha C \]

   \textbf{Note:} This example uses the result of the last answer to perform computations on the TI-89 Titanium. This feature reduces keystroking and chances for error.
4. Divide both sides of the equation by the leading coefficient \( a \).

**Note:** Continue to use the last answer \((2\text{nd} \text{[ANS]})\) as in step 3 in steps 4 through 9.

5. Use the `expand()` function to expand the result of the last answer.

6. Complete the square by adding \(((b/a)/2)^2\) to both sides of the equation.

7. Factor the result using the `factor()` function.

8. Multiply both sides of the equation by \(4a^2\).
9. Take the square root of both sides of the equation with the constraint that \( a > 0 \) and \( b > 0 \) and \( x > 0 \).

10. Solve for \( x \) by subtracting \( b \) from both sides and then dividing by \( 2a \).

**Note:** This is only one of the two general quadratic solutions due to the constraint in step 9.

**Exploring a Matrix**

This activity shows you how to perform several matrix operations.
Exploring a 3x3 Matrix

Perform these steps to generate a random matrix, augment and find the identity matrix, and then solve to find an invalid value of the inverse.

1. On the Home screen, use **RandSeed** to set the random number generator seed to the factory default, and then use **randMat( )** to create a random 3x3 matrix and store it in **a**.

2. Replace the [2,3] element of the matrix with the variable **x**, and then use the **augment( )** function, to augment the 3x3 identity to **a** and store the result in **b**.

3. Use **rref( )** to “row reduce” matrix **b**:

   The result will have the identity matrix in the first three columns and **a^-1** in the last three columns.

   **Note:** Use the cursor in the history area to scroll the result.
4. Solve for the value of $x$ that will cause the inverse of the matrix to be invalid.

Enter:
\[
\text{solve(getDenom( 2nd [ANS] [1,4] )=0,x)}
\]
Result: $x = -70/17$

Note: Use the cursor in the history area to scroll the result.

Exploring $\cos(x) = \sin(x)$

This activity uses two methods to find where $\cos(x) = \sin(x)$ for the values of $x$ between 0 and $3\pi$.

Method 1: Graph Plot

Perform the following steps to observe where the graphs of the functions $y_1(x)=\cos(x)$ and $y_2(x)=\sin(x)$ intersect.

1. In the Y= Editor, set $y_1(x)=\cos(x)$ and $y_2(x)=\sin(x)$.
2. In the Window Editor, set $\text{xmin}=0$ and $\text{xmax}=3\pi$.
3. Press $\text{F2}$ and select A:ZoomFit.
4. Find the intersection point of the two functions.

**Note:** Press [F5] and select 5:Intersection. Respond to the screen prompts to select the two curves, and the lower and upper bounds for intersection A.

5. Note the $x$ and $y$ coordinates. (Repeat steps 4 and 5 to find the other intersections.)

### Method 2: Symbolic Manipulation

Perform the following steps to solve the equation $\sin(x) = \cos(x)$ with respect to $x$.

1. On the Home screen, enter `solve(sin(x)=cos(x),x)`.
   
   The solution for $x$ is where $@n1$ is any integer.

2. Using the `ceiling()` and `floor()` functions, find the ceiling and floor values for the intersection points as shown.

   **Note:** Move the cursor into the history area to highlight the last answer. Press [ENTER] to copy the result of the general solution.
Finding Minimum Surface Area of a Parallelepiped

This activity shows you how to find the minimum surface area of a parallelepiped having a constant volume $V$. Detailed information about the steps used in this example can be found in Symbolic Manipulation and 3D Graphing.

Exploring a 3D Graph of the Surface Area of a Parallelepiped

Perform the following steps to define a function for the surface area of a parallelepiped, draw a 3D graph, and use the Trace tool to find a point close to the minimum surface area.

1. On the Home screen, define the function $sa(x,y,v)$ for the surface area of a parallelepiped.

   Enter: `define sa(x,y,v)=2*x*y + 2v/x+2v/y`

3. Enter the general solution for $x$ and apply the constraint for $\texttt{@n1}$ as shown. Compare the result with Method 1.

Note: To get the with operator, press:  

![Image of calculation steps]
2. Select the 3D Graph mode. Then enter the function for $z_1(x,y)$ as shown in this example with volume $v=300$.

3. Set the Window variables to:
   
   $\text{eye} = [60,90,0]$

   $\text{x} = [0,15,15]$

   $\text{y} = [0,15,15]$

   $\text{z} = [260,300]$

   $\text{ncontour} = 5$

4. Graph the function and use Trace to go to the point close to the minimum value of the surface area function.

**Finding the Minimum Surface Area Analytically**

Perform the following steps to solve the problem analytically on the Home screen.

1. Solve for $x$ and $y$ in terms of $v$.
   
   Enter: $\text{solve(}d(\text{sa}(x,y,v),x)=0 \text{ and } d(\text{sa}(x,y,v),y)=0,\{x,y\})$
2. Find the minimum surface area when the value of $v$ equals 300.

Enter: $300 \rightarrow v$

Enter: $sa(v^{\frac{1}{3}}, v^{\frac{1}{3}}, v)$

**Note:** Press [ENTER] to obtain the exact result in symbolic form. Press $\uparrow$ [ENTER] to obtain the approximate result in decimal form.

---

**Running a Tutorial Script Using the Text Editor**

This activity shows you how to use the **Text Editor** to run a tutorial script.

**Running a Tutorial Script**

Perform the following steps to write a script using the **Text Editor**, test each line, and observe the results in the history area on the Home screen.

1. Open the **Text Editor**, and create a new variable named **demo1**.

   **Note:** The command symbol **C** is accessed from the $F2$ **1:Command** toolbar menu.
2. Type the following lines into the Text Editor.

\[
\text{: Compute the maximum value of } f \text{ on the closed interval } [a,b] \\
\text{: assume that } f \text{ is differentiable on } [a,b] \\
\text{C : define } f(x) = x^3 - 2x^2 + x - 7 \\
\text{C : } 1 \rightarrow a : 3.22 \rightarrow b \\
\text{C : } d(f(x),x) \rightarrow df(x) \\
\text{C : } \text{zeros}(df(x),x) \\
\text{C : } f(\text{ans}(1)) \\
\text{C : } f(\{a,b\})
\]

: The largest number from the previous two commands is the maximum value of the function. The smallest number is the minimum value.

3. Press \[F3\] and select 1:Script view to show the Text Editor and the Home screen on a split-screen. Move the cursor to the first line in the Text Editor.
4. Press $\text{F4}$ repeatedly to execute each line in the script one at a time.

**Note:** Press $\text{F4}$ and select 2:Clear split to go back to a full-sized Text Editor screen.

5. To see the results of the script on a full-sized screen, go to the Home screen.

**Note:** Press $\text{2nd [QUIT]}$ twice to display the Home screen.

**Decomposing a Rational Function**

This activity examines what happens when a rational function is decomposed into a quotient and remainder. Detailed information about the steps used in this example can be found in *Basic Function Graphing* and *Symbolic Manipulation*. 
Decomposing a Rational Function

To examine the decomposition of the rational function \( f(x) = \frac{x^3 - 10x^2 - x + 50}{x - 2} \) on a graph:

1. On the Home screen, enter the rational function as shown below and store it in a function \( f(x) \).
   Enter: \( (x^3 - 10x^2 - x + 50)/(x - 2) \) > f(x)
   **Note:** Actual entries are displayed in reverse type in the example screens.

2. Use the proper fraction function (propFrac) to split the function into a quotient and remainder.

3. Copy the last answer to the entry line.
   –or–
   Enter: \( 16/(x - 2) + x^2 - 8x - 17 \)
   **Note:** Move the cursor into the history area to highlight the last answer. Press [ENTER] to copy it to the entry line.

4. Edit the last answer in the entry line. Store the remainder to \( y_1(x) \) and the quotient to \( y_2(x) \) as shown.
   Enter: \( 16/(x - 2) \) > \( y_1(x) \): \( x^2 - 8x - 17 \) > \( y_2(x) \)
5. In the **Y= Editor**, select the thick graphing style for $y_2(x)$.

6. Add the original function $f(x)$ to $y_3(x)$ and select the square graphing style.

7. In the **Window Editor**, set the window variables to:

   - $x = [-10, 15, 10]$
   - $y = [-100, 100, 10]$

8. Draw the graph.

   **Note:** Be sure the Graph mode is set to Function.
Observe that the global behavior of the $f(x)$ function is basically represented by the quadratic quotient $y_2(x)$. The rational expression is basically a quadratic function as $x$ gets very large in both the positive and negative directions.

The lower graph is $y_3(x)=f(x)$ graphed separately using the line style.

**Studying Statistics: Filtering Data by Categories**

This activity provides a statistical study of the weights of high school students using categories to filter the data.
Filtering Data by Categories

Each student is placed into one of eight categories depending on the student’s sex and academic year (freshman, sophomore, junior, or senior). The data (weight in pounds) and respective categories are entered in the Data/Matrix Editor.

### Table 1: Category vs. Description

<table>
<thead>
<tr>
<th>Category (C2)</th>
<th>Academic Year and Sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Freshman boys</td>
</tr>
<tr>
<td>2</td>
<td>Freshman girls</td>
</tr>
<tr>
<td>3</td>
<td>Sophomore boys</td>
</tr>
<tr>
<td>4</td>
<td>Sophomore girls</td>
</tr>
<tr>
<td>5</td>
<td>Junior boys</td>
</tr>
<tr>
<td>6</td>
<td>Junior girls</td>
</tr>
<tr>
<td>7</td>
<td>Senior boys</td>
</tr>
<tr>
<td>8</td>
<td>Senior girls</td>
</tr>
</tbody>
</table>

### Table 2: C1 (weight of each student in pounds) vs. C2 (category)

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C1</th>
<th>C2</th>
<th>C1</th>
<th>C2</th>
<th>C1</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>1</td>
<td>115</td>
<td>3</td>
<td>130</td>
<td>5</td>
<td>145</td>
<td>7</td>
</tr>
<tr>
<td>125</td>
<td>1</td>
<td>135</td>
<td>3</td>
<td>145</td>
<td>5</td>
<td>160</td>
<td>7</td>
</tr>
<tr>
<td>105</td>
<td>1</td>
<td>110</td>
<td>3</td>
<td>140</td>
<td>5</td>
<td>165</td>
<td>7</td>
</tr>
<tr>
<td>120</td>
<td>1</td>
<td>130</td>
<td>3</td>
<td>145</td>
<td>5</td>
<td>170</td>
<td>7</td>
</tr>
<tr>
<td>140</td>
<td>1</td>
<td>150</td>
<td>3</td>
<td>165</td>
<td>5</td>
<td>190</td>
<td>7</td>
</tr>
<tr>
<td>85</td>
<td>2</td>
<td>90</td>
<td>4</td>
<td>100</td>
<td>6</td>
<td>110</td>
<td>8</td>
</tr>
<tr>
<td>80</td>
<td>2</td>
<td>95</td>
<td>4</td>
<td>105</td>
<td>6</td>
<td>115</td>
<td>8</td>
</tr>
<tr>
<td>90</td>
<td>2</td>
<td>85</td>
<td>4</td>
<td>115</td>
<td>6</td>
<td>125</td>
<td>8</td>
</tr>
<tr>
<td>80</td>
<td>2</td>
<td>100</td>
<td>4</td>
<td>110</td>
<td>6</td>
<td>120</td>
<td>8</td>
</tr>
<tr>
<td>95</td>
<td>2</td>
<td>95</td>
<td>4</td>
<td>120</td>
<td>6</td>
<td>125</td>
<td>8</td>
</tr>
</tbody>
</table>
Perform the following steps to compare the weight of high school students to their year in school.

1. Start the **Data/Matrix Editor**, and create a new Data variable named **students**.

2. Enter the data and categories from Table 2 into columns **c1** and **c2**, respectively.

3. Open the **F2 Plot Setup** toolbar menu. **Note:** Set up several box plots to compare different subsets of the entire data set.

4. Define the plot and filter parameters for **Plot 1** as shown in this screen.
5. Copy Plot 1 to Plot 2.

6. Repeat step 5 and copy Plot 1 to Plot 3, Plot 4, and Plot 5.

7. Press \( F_1 \), and modify the Include Categories item for Plot 2 through Plot 5 to the following:
   
   Plot 2: \{1,2\}
   (freshman boys, girls)
   Plot 3: \{7,8\}
   (senior boys, girls)
   Plot 4: \{1,3,5,7\}
   (all boys)
   Plot 5: \{2,4,6,8\}
   (all girls)

8. In the \( Y= \) Editor, deselect any functions that may be selected from a previous activity.

   Note: Only Plot 1 through Plot 5 should be selected.
This activity provides a program that can be used when the TI-89 Titanium is connected to a Calculator-Based Laboratory™ (CBL 2™) unit. This program works with the “Newton’s Law of Cooling” experiment. You can use your computer keyboard to type lengthy text and then use TI Connect™ software to send it to the calculator. More CBL 2™ programs are available from the TI Web site at educaton.ti.com.

<table>
<thead>
<tr>
<th>Program Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:cooltemp( )</td>
<td>Program name</td>
</tr>
<tr>
<td>:Prgm</td>
<td></td>
</tr>
<tr>
<td>:Local i</td>
<td>Declare local variable; exists only at run time.</td>
</tr>
</tbody>
</table>

**CBL 2™ Program for the TI-89 Titanium**

9. Display the plots by pressing [F2] and selecting **9:ZoomData**.

10. Use the **Trace** tool to compare the median student weights for different subsets.

   1. median, all students
   2. all students
   3. all freshmen
   4. all seniors
   5. all boys
   6. all girls
### Program Instruction and Description

<table>
<thead>
<tr>
<th>Program Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:setMode(&quot;Graph&quot;,&quot;FUNCTION&quot;)</td>
<td>Set up the TI-89 Titanium for function graphing.</td>
</tr>
<tr>
<td>:PlotsOff</td>
<td>Turn off any previous plots.</td>
</tr>
<tr>
<td>:FnOff</td>
<td>Turn off any previous functions.</td>
</tr>
<tr>
<td>:ClrDraw</td>
<td>Clear any items previously drawn on graph screens.</td>
</tr>
<tr>
<td>:ClrGraph</td>
<td>Clear any previous graphs.</td>
</tr>
<tr>
<td>:ClrIO</td>
<td>Clear the TI-89 Titanium Program IO (input/output) screen.</td>
</tr>
<tr>
<td>:-10→xmin:99→xmax:10→xscl</td>
<td>Set up the Window variables.</td>
</tr>
<tr>
<td>:-20→ymin:100→ymax:10→yscl</td>
<td></td>
</tr>
<tr>
<td>:{0}→data</td>
<td>Create and/or clear a list named data.</td>
</tr>
<tr>
<td>:{0}→time</td>
<td>Create and/or clear a list named time.</td>
</tr>
<tr>
<td>:Send{1,0}</td>
<td>Send a command to clear the CBL 2™ unit.</td>
</tr>
<tr>
<td>:Send{1,2,1}</td>
<td>Set up Chan. 2 of the CBL 2™ to AutoID to record temp.</td>
</tr>
<tr>
<td>:Disp &quot;Press ENTER to start&quot;</td>
<td>Prompt the user to press [ENTER].</td>
</tr>
<tr>
<td>:Disp &quot;graphingTemperature.&quot;</td>
<td></td>
</tr>
<tr>
<td>:Pause</td>
<td>Wait until the user is ready to start.</td>
</tr>
<tr>
<td>:PtText &quot;TEMP(C)&quot;,2,99</td>
<td>Label the y axis of the graph.</td>
</tr>
<tr>
<td>:PtText &quot;T(S)&quot;,80,−5</td>
<td>Label the x axis of the graph.</td>
</tr>
</tbody>
</table>
You can also use the Calculator-Based Ranger™ system (CBR™) to explore the mathematical and scientific relationships between distance, velocity, acceleration, and time using data collected from activities you perform.

### Studying the Flight of a Hit Baseball

This activity uses the split screen settings to show a parametric graph and a table at the same time to study the flight of a hit baseball.

<table>
<thead>
<tr>
<th>Program Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:Send{3,1,-1,0}</td>
<td>Send the Trigger command to the CBL 2™; collect data in real-time.</td>
</tr>
<tr>
<td>:For i,1,99</td>
<td>Repeat next two instructions for 99 temperature readings.</td>
</tr>
<tr>
<td>:Get data[i]</td>
<td>Get a temperature from the CBL 2™ and store it in a list.</td>
</tr>
<tr>
<td>:PtOn i,data[i]</td>
<td>Plot the temperature data on a graph.</td>
</tr>
<tr>
<td>:EndFor</td>
<td></td>
</tr>
<tr>
<td>:seq(i,i,1,99,1)→time</td>
<td>Create a list to represent time or data sample number.</td>
</tr>
<tr>
<td>:NewPlot 1,1,time,data,,,4</td>
<td>Plot time and data using NewPlot and the Trace tool.</td>
</tr>
<tr>
<td>:DispG</td>
<td>Display the graph.</td>
</tr>
<tr>
<td>:PtText &quot;TEMP(C)&quot;,2,99</td>
<td>Re-label the axes.</td>
</tr>
<tr>
<td>:PtText &quot;T(S)&quot;,80,-5</td>
<td></td>
</tr>
<tr>
<td>:EndPrgm</td>
<td>Stop the program.</td>
</tr>
</tbody>
</table>
Setting Up a Parametric Graph and Table

Perform the following steps to study the flight of a hit baseball that has an initial velocity of 95 feet per second and an initial angle of 32 degrees.

1. Set the modes for **Page 1** as shown in this screen.

2. Set the modes for **Page 2** as shown in this screen.

3. In the **Y= Editor** on the left side, enter the equation for the distance of the ball at time \( t \) for \( xt1(t) \).

   \[
   xt1(t)=95*t*cos(32^\circ)
   \]

   **Note:** Press \[2^{nd}[\ ^{\circ}\] to obtain the degree symbol.

4. In the **Y= Editor**, enter the equation for the height of the ball at time \( t \) for \( yt1(t) \).

   \[
   yt1(t)=-16*t^2+95*t*sin(32^\circ)
   \]
5. Set the Window variables to:
   \[
   \begin{align*}
   t \text{ values} &= [0,4,.1] \\
   x \text{ values} &= [0,300,50] \\
   y \text{ values} &= [0,100,10]
   \end{align*}
   \]

6. Switch to the right side and display the graph.
   **Note:** Press \(2\text{nd} [\pm]\).

7. Display the **TABLE SETUP** dialog box, and change \(\text{tblStart}\) to 0 and \(\Delta\text{tbl}\) to 0.1.
   **Note:** Press \(\uparrow [\text{TBLSET}]\).

8. Display the table in the left side and press \(\uparrow\) to highlight \(t=2\).
   **Note:** Press \(\uparrow [\text{TABLE}]\).

9. Switch to the right side. Press \(\text{F3}\), and trace the graph to show the values of \(x_c\) and \(y_c\) when \(t_c=2\).
   **Note:** As you move the trace cursor from \(t_c=0.0\) to \(t_c=3.1\), you will see the position of the ball at time \(t_c\).
Optional Exercise

Assuming the same initial velocity of 95 feet per second, find the angle that the ball should be hit to achieve the greatest distance.

Visualizing Complex Zeros of a Cubic Polynomial

This activity describes graphing the complex zeros of a cubic polynomial.

Visualizing Complex Roots

Perform the following steps to expand the cubic polynomial \((x-1)(x-i)(x+i)\), find the absolute value of the function, graph the modulus surface, and use the Trace tool to explore the modulus surface.

1. On the Home screen, use the \texttt{expand()} function to expand the cubic expression \((x-1)(x-i)(x+i)\) and see the first polynomial.

2. Copy and paste the last answer to the entry line and store it in the function \(f(x)\).

   \textbf{Note:} Move the cursor into the history area to highlight the last answer and press \texttt{ENTER}, to copy it to the entry line.
3. Use the \texttt{abs()} function to find the absolute value of \( f(x+y) \).

(This calculation may take about 2 minutes.)

\textbf{Note:} The absolute value of a function forces any roots to visually just touch rather than cross the \( x \) axis. Likewise, the absolute value of a function of two variables will force any roots to visually just touch the \( xy \) plane.

4. Copy and paste the last answer to the entry line and store it in the function \( z_1(x,y) \).

\textbf{Note:} The graph of \( z_1(x,y) \) will be the modulus surface.

5. Set the unit to 3D graph mode, turn on the axes for graph format, and set the Window variables to:

\begin{align*}
\text{eye=} & \quad [20,70,0] \\
\text{x=} & \quad [-2,2,20] \\
\text{y=} & \quad [-2,2,20] \\
\text{z=} & \quad [-1,2] \\
\text{ncontour=} & \quad [5]
\end{align*}
6. In the Y=Editor, press:

and set the Graph Format variables to:

- **Axes**: ON
- **Labels**: ON
- **Style**: HIDDEN SURFACE

**Note**: Calculating and drawing the graph takes about three minutes.

7. Graph the modulus surface.

The 3D graph is used to visually display a picture of the roots where the surface touches the \(xy\) plane.

8. Use the Trace tool to explore the function values at \(x=1\) and \(y=0\).

9. Use the Trace tool to explore the function values at \(x=0\) and \(y=1\).
10. Use the Trace tool to explore the function values at \(x=0\) and \(y=-1\).

**Summary**

Note that \(zc\) is zero for each of the function values in steps 7–9. Thus, the complex zeros \(1, -i, i\) of the polynomial \(x^3 - x^2 + x - 1\) can be visualized with the three points where the graph of the modulus surface touches the \(xy\) plane.

**Solving a Standard Annuity Problem**

This activity can be used to find the interest rate, starting principal, number of compounding periods, and future value of an annuity.
Finding the Interest Rate of an Annuity

Perform the following steps to find the interest rate \( (i) \) of an annuity where the starting principal \( (p) \) is 1,000, number of compounding periods \( (n) \) is 6, and the future value \( (s) \) is 2,000.

1. On the Home screen, enter the equation to solve for \( p \).

   \[
   \text{solve} \left( s = p \cdot (1 + i)^n, p \right)
   \]

   \[
   p = (i + 1)^{-n} \cdot s
   \]

2. Enter the equation to solve for \( n \).

   \[
   \text{solve} \left( s = p \cdot (1 + i)^n, n \right)
   \]

   \[
   n = \frac{\ln \left( \frac{s}{p} \right)}{\ln (1 + i)} \quad \text{and} \quad \frac{s}{p} > 0
   \]

3. Enter the equation to solve for \( i \) using the \textit{with} operator.

   \[
   \text{solve} \left( s = p \cdot (1 + i)^n, i \right) \mid s=2000 \quad \text{and} \quad p=1000 \quad \text{and} \quad n=6
   \]

   Result: The interest rate is 12.246%.

   \textbf{Note:}

   • To enter the \textit{“with”} ( | ) operator: 

   • Press \( \downarrow \) \textbf{ENTER} to obtain a floating-point result.
**Finding the Future Value of an Annuity**

Find the future value of an annuity using the values from the previous example where the interest rate is 14%.

Enter the equation to solve for \( s \).

\[
\text{solve}(s=p*(1+i)^n,s) \mid i=.14 \text{ and } p=1000 \text{ and } n=6
\]

Result: The future value at 14% interest is 2,194.97.

**Computing the Time-Value-of-Money**

This activity creates a function that can be used to find the cost of financing an item. Detailed information about the steps used in this example can be found in the electronic chapter *Programming*, which is available from the TI Web site at education.ti.com and on the CD in this package.

**Time-Value-of-Money Function**

In the Program Editor, define the following Time-Value-of-Money (tvm) function where \( \text{temp1} = \) number of payments, \( \text{temp2} = \) annual interest rate, \( \text{temp3} = \) present value, \( \text{temp4} = \) monthly payment, \( \text{temp5} = \) future value, and \( \text{temp6} = \) begin- or end-of-payment
period (1 = beginning of month, 0 = end of month).

:tvm(temp1,temp2,temp3,temp4,temp5,temp6)
:Func
:Local tempi,tempfunc,tempstr1
:-temp3+(1+temp2/1200 temp6) temp4 ((1−(1+temp2/1200)^(-temp1)) \((\text{temp2}/1200))−temp5 (1+temp2/1200)^{-\text{temp1}}
→tempfunc
:For tempi,1,5,1
:"temp"&exact(string(tempi)→tempstr1
:If when(#tempstr1=0,false,false,true) Then
:If tempi=2
:Return approx(nsolve(tempfunc=0,#tempstr1) | #tempstr1>0 and #tempstr1<100)
:Return approx(nsolve(tempfunc=0,#tempstr1))
:EndIf
:EndFor
:Return "parameter error"
:EndFunc

Note: You can use your computer keyboard to type lengthy text and then use TI Connect™ software to send it to the TI-89 Titanium.
Finding the Monthly Payment

Find the monthly payment on 10,000 if you make 48 payments at 10% interest per year.

On the Home screen, enter the \texttt{tvm} values to find \texttt{pmt}.

Result: The monthly payment is 251.53.

Finding the Number of Payments

Find the number of payments it will take to pay off the loan if you could make a 300 payment each month.

On the Home screen, enter the \texttt{tvm} values to find \texttt{n}.

Result: The number of payments is 38.8308.

Finding Rational, Real, and Complex Factors

This activity shows how to find rational, real, or complex factors of expressions. Detailed information about the steps used in this example can be found in \textit{Symbolic Manipulation}. 
Finding Factors

Enter the expressions shown below on the Home screen.

1. \( \text{factor}(x^3 - 5x) \) \( \text{ENTER} \) displays a rational result.

2. \( \text{factor}(x^3 + 5x) \) \( \text{ENTER} \) displays a rational result.

3. \( \text{factor}(x^3 - 5x, x) \) \( \text{ENTER} \) displays a real result.

4. \( \text{cfactor}(x^3 + 5x, x) \) \( \text{ENTER} \) displays a complex result.
Simulation of Sampling without Replacement

This activity simulates drawing different colored balls from an urn without replacing them. Detailed information about the steps used in this example can be found in the electronic chapter *Programming*.

**Sampling-without- Replacement Function**

In the **Program Editor**, define **drawball( )** as a function that can be called with two parameters. The first parameter is a list where each element is the number of balls of a certain color. The second parameter is the number of balls to select. This function returns a list where each element is the number of balls of each color that were selected.

```plaintext
:drawball(urnlist,drawnum)
:Func
:Local templist,drawlist,colordim,
    numballels,i,pick,urncum,j
:If drawnum>sum(urnlist)
:Return “too few balls”
:dim(urnlist)⇒colordim
:urnlist⇒templist
:newlist(colordim)⇒drawlist
:For i,1,drawnum,1
:sum(templist)⇒numballs
:rand(numballs)⇒pick
:For j,1,colordim,1
:cumSum(templist)⇒urncum
:If pick ≤ urncum[j] Then
:drawlist[j]+1⇒drawlist[j]
:templist[j]–1⇒templist[j]
:Exit
:EndIf
:EndFor
:EndFor
:Return drawlist
:EndFunc
```

(continued in next column)
Sampling without Replacement

Suppose an urn contains $n_1$ balls of a color, $n_2$ balls of a second color, $n_3$ balls of a third color, etc. Simulate drawing balls without replacing them.

1. Enter a random seed using the **RandSeed** command.

2. Assuming the urn contains 10 red balls and 25 white balls, simulate picking 5 balls at random from the urn without replacement. Enter **drawball({10,25},5)**.
   
   Result: 2 red balls and 3 white balls.

Using Vectors to Determine Velocity

A small fishing boat leaves from the south bank of the Allegheny River and heads at an 80° angle with an engine speed of 20 knots. However, the eastward force of the current carries the boat along so it actually travels at a 60° angle with the shore.

How fast is the current, and how fast does the boat actually travel?
1. Set the modes for **Page 1** as shown in this screen. (Show angles in degrees and display all digits with a floating decimal point.)


2. Set the modes for **Page 2** as shown in this screen. (Display answers in decimal form.)

   Press: **MODE** [F2] + + . On the Exact/Approx option, select **3:APPROXIMATE**.
3. Enter vectors describing the initial path of the boat, water current, and resultant path of the boat.

Store these vectors as \( \mathbf{i}, \mathbf{c}, \) and \( \mathbf{r}. \) Use the value \( a \) for the unknown speed of the current. Use the value \( b \) for the speed of the boat.

Enter:

\[
\begin{align*}
[20, 80^\circ] & \rightarrow \mathbf{i} \\
[a, 0^\circ] & \rightarrow \mathbf{c} \\
[b, 60^\circ] & \rightarrow \mathbf{r}
\end{align*}
\]

Vectors are commonly written in either polar or rectangular form, so it is useful to convert polar vectors into rectangular form.

4. Define function \( p2r. \)

Enter: \( \text{Define } p2r(x) = [x[1,1] \cos(x[1,2]), x[1,1] \sin(x[1,2])] \)
When converted to rectangular form, the sum of vectors \( \mathbf{i} \) and \( \mathbf{c} \) equals the resultant vector \( \mathbf{r} \).

5. Using function \( \text{p2r} \), convert vectors \( \mathbf{i} \), \( \mathbf{c} \), and \( \mathbf{r} \) to rectangular form.

Enter:

\[
\begin{align*}
p2r(i) & \to i \\
p2r(c) & \to c \\
p2r(r) & \to r
\end{align*}
\]

Because the vectors are equal, the x-coordinate of \( \mathbf{i} + \mathbf{c} \) must equal the x-coordinate of the resultant vector \( \mathbf{r} \). Likewise, the y-coordinate of \( \mathbf{i} + \mathbf{c} \) must equal the y-coordinate of resultant vector \( \mathbf{r} \).

6. Set up two equations involving vectors \( \mathbf{i} + \mathbf{c} \) and \( \mathbf{r} \).

- Equation 1 sets the x-coordinates equal to each other.
- Equation 2 sets the y-coordinates equal.

Store these equations into \( \text{eq1} \) and \( \text{eq2} \), respectively. Enter:

\[
\begin{align*}
i[1,1] + c[1,1] & = r[1,1] \to \text{eq1} \\
i[1,2] + c[1,2] & = r[1,2] \to \text{eq2}
\end{align*}
\]
7. Solve \( eq2 \) for \( b \) to calculate the actual speed of the boat.

\[
\text{solve}(eq2,b)
\]

8. Substitute the known value of \( b \) into \( eq1 \), and solve \( eq1 \) for \( a \) to determine \( a \), the speed of the eastward traveling current.

\[
\text{solve}(eq1,a) \mid b
\]

The boat travels at a speed of 22.7 knots, and the water current is approximately 7.9 knots.
Appendix A: Functions and Instructions

This section describes the syntax and action of each TI-89 Titanium function and instruction that is included in the operating system (OS). See modules relating to calculator software applications (Apps) for functions and instructions specific to those Apps.

Name of the function or instruction.
Key or menu for entering the name.
You can also type the name.

**Circle**

**CATALOG**

Draws a circle with its center at window coordinates \((x, y)\) and with a radius of \(r\).

- \(x, y, r\): arguments must be real values.
- If \(\text{drawMode} = 1\), draws the circle (default).
- If \(\text{drawMode} = 0\), turns off the circle.
- If \(\text{drawMode} = -1\), inverts pixels along the circle.

**Note**: Regraphing erases all drawn items.

In a ZoomSqr viewing window:

```
ZoomSqr:Circle 1.2,3
```

Syntax line shows the order and the type of arguments that you supply. Be sure to separate multiple arguments with a comma (,).

Arguments are shown in **italics**.
Arguments in [ ] brackets are optional. Do not type the brackets.

Example

Exam

ple

Explanation of the function or instruction.
## Categorical Listing of Operations

This section lists the TI-89 Titanium functions and instructions in functional groups along with the page numbers where they are described.

### Algebra

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>factor()</code></td>
<td>819</td>
</tr>
<tr>
<td><code>cSolve()</code></td>
<td>819</td>
</tr>
<tr>
<td><code>cZeros</code></td>
<td>803</td>
</tr>
<tr>
<td><code>getDenom()</code></td>
<td>825</td>
</tr>
<tr>
<td><code>comDenom()</code></td>
<td>817</td>
</tr>
<tr>
<td><code>randPoly()</code></td>
<td>862</td>
</tr>
<tr>
<td><code>propFrac()</code></td>
<td>856</td>
</tr>
<tr>
<td><code>expand()</code></td>
<td>817</td>
</tr>
<tr>
<td><code>getNum()</code></td>
<td>825</td>
</tr>
<tr>
<td><code>nSolve()</code></td>
<td>849</td>
</tr>
<tr>
<td><code>zeros()</code></td>
<td>895</td>
</tr>
<tr>
<td><code>tExpand()</code></td>
<td>889</td>
</tr>
<tr>
<td><code>tCollect()</code></td>
<td>888</td>
</tr>
</tbody>
</table>

### Calculus

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>integrate()</code></td>
<td>907</td>
</tr>
<tr>
<td><code>product()</code></td>
<td>908</td>
</tr>
<tr>
<td><code>sum()</code></td>
<td>908</td>
</tr>
<tr>
<td><code>arcLen()</code></td>
<td>788</td>
</tr>
<tr>
<td><code>avgRC()</code></td>
<td>790</td>
</tr>
<tr>
<td><code>d()</code></td>
<td>805</td>
</tr>
<tr>
<td><code>fMax()</code></td>
<td>821</td>
</tr>
<tr>
<td><code>fMin()</code></td>
<td>821</td>
</tr>
<tr>
<td><code>seq()</code></td>
<td>856</td>
</tr>
<tr>
<td><code>nDeriv()</code></td>
<td>869</td>
</tr>
<tr>
<td><code>taylor()</code></td>
<td>888</td>
</tr>
</tbody>
</table>

### Graphics

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AndPic</code></td>
<td>787</td>
</tr>
<tr>
<td><code>BldData</code></td>
<td>790</td>
</tr>
<tr>
<td><code>CyclePic</code></td>
<td>803</td>
</tr>
<tr>
<td><code>ClrGraph</code></td>
<td>793</td>
</tr>
<tr>
<td><code>DrawInv</code></td>
<td>812</td>
</tr>
<tr>
<td><code>DrawParm</code></td>
<td>813</td>
</tr>
<tr>
<td><code>LineHorz</code></td>
<td>835</td>
</tr>
<tr>
<td><code>LineTan</code></td>
<td>836</td>
</tr>
<tr>
<td><code>LineVert</code></td>
<td>846</td>
</tr>
<tr>
<td><code>PtChg</code></td>
<td>856</td>
</tr>
<tr>
<td><code>PtOff</code></td>
<td>856</td>
</tr>
<tr>
<td><code>PtOn</code></td>
<td>856</td>
</tr>
<tr>
<td><code>PxlChg</code></td>
<td>857</td>
</tr>
<tr>
<td><code>PxlCrl</code></td>
<td>857</td>
</tr>
<tr>
<td><code>PxlLine</code></td>
<td>857</td>
</tr>
<tr>
<td><code>PxlOff</code></td>
<td>858</td>
</tr>
<tr>
<td><code>PxlTest()</code></td>
<td>858</td>
</tr>
<tr>
<td><code>PxlText</code></td>
<td>858</td>
</tr>
<tr>
<td><code>RcI GórD</code></td>
<td>862</td>
</tr>
<tr>
<td><code>RcIPic</code></td>
<td>862</td>
</tr>
<tr>
<td><code>Shade</code></td>
<td>874</td>
</tr>
<tr>
<td><code>StoGDB</code></td>
<td>883</td>
</tr>
<tr>
<td><code>Style</code></td>
<td>884</td>
</tr>
<tr>
<td><code>Trace</code></td>
<td>891</td>
</tr>
<tr>
<td><code>ZoomBox</code></td>
<td>896</td>
</tr>
<tr>
<td><code>ZoomData</code></td>
<td>897</td>
</tr>
<tr>
<td><code>ZoomFit</code></td>
<td>898</td>
</tr>
<tr>
<td><code>ZoomIn</code></td>
<td>898</td>
</tr>
<tr>
<td><code>ZoomOut</code></td>
<td>899</td>
</tr>
<tr>
<td><code>ZoomPrev</code></td>
<td>899</td>
</tr>
<tr>
<td><code>ZoomStd</code></td>
<td>900</td>
</tr>
<tr>
<td><code>ZoomTrig</code></td>
<td>900</td>
</tr>
</tbody>
</table>

### Lists

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>add()</code></td>
<td>900</td>
</tr>
<tr>
<td><code>subtract</code></td>
<td>901</td>
</tr>
<tr>
<td><code>multiply</code></td>
<td>902</td>
</tr>
<tr>
<td><code>negate</code></td>
<td>903</td>
</tr>
<tr>
<td><code>augment()</code></td>
<td>789</td>
</tr>
<tr>
<td><code>crossP()</code></td>
<td>798</td>
</tr>
<tr>
<td><code>cumSum()</code></td>
<td>802</td>
</tr>
<tr>
<td><code>dotP()</code></td>
<td>812</td>
</tr>
<tr>
<td><code>expList()</code></td>
<td>817</td>
</tr>
<tr>
<td><code>list()</code></td>
<td>837</td>
</tr>
<tr>
<td><code>listMat()</code></td>
<td>837</td>
</tr>
<tr>
<td><code>max()</code></td>
<td>842</td>
</tr>
<tr>
<td><code>mid()</code></td>
<td>843</td>
</tr>
<tr>
<td><code>newList()</code></td>
<td>846</td>
</tr>
<tr>
<td><code>polyEval()</code></td>
<td>854</td>
</tr>
<tr>
<td><code>right()</code></td>
<td>865</td>
</tr>
<tr>
<td><code>rotate()</code></td>
<td>865</td>
</tr>
<tr>
<td><code>sortA</code></td>
<td>881</td>
</tr>
<tr>
<td><code>sortD</code></td>
<td>881</td>
</tr>
</tbody>
</table>

---

Appendix A: Functions and Instructions
### Math

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ (add)</td>
<td>900</td>
</tr>
<tr>
<td>- (subtract)</td>
<td>901</td>
</tr>
<tr>
<td>* (multiply)</td>
<td>902</td>
</tr>
<tr>
<td>/ (divide)</td>
<td>902</td>
</tr>
<tr>
<td>` (negate)</td>
<td>904</td>
</tr>
<tr>
<td>% (percent)</td>
<td>904</td>
</tr>
<tr>
<td>! (factorial)</td>
<td>906</td>
</tr>
<tr>
<td>√() (sqr. root)</td>
<td>908</td>
</tr>
<tr>
<td>^ (power)</td>
<td>903</td>
</tr>
<tr>
<td>° (degree)</td>
<td>909</td>
</tr>
<tr>
<td>′ (angle)</td>
<td>909</td>
</tr>
<tr>
<td>÷ (underscore)</td>
<td>910</td>
</tr>
<tr>
<td>+ (add)</td>
<td>911</td>
</tr>
<tr>
<td>- (subtract)</td>
<td>913</td>
</tr>
<tr>
<td>* (multiply)</td>
<td>911</td>
</tr>
</tbody>
</table>

#### Matrices

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ (add)</td>
<td>900</td>
</tr>
<tr>
<td>- (subtract)</td>
<td>901</td>
</tr>
<tr>
<td>* (multiply)</td>
<td>902</td>
</tr>
<tr>
<td>/ (divide)</td>
<td>902</td>
</tr>
<tr>
<td>` (negate)</td>
<td>904</td>
</tr>
<tr>
<td>. (dot mult.)</td>
<td>904</td>
</tr>
<tr>
<td>.` (dot divide)</td>
<td>904</td>
</tr>
<tr>
<td>.^ (power)</td>
<td>904</td>
</tr>
<tr>
<td>colDim()</td>
<td>793</td>
</tr>
<tr>
<td>colNorm()</td>
<td>793</td>
</tr>
<tr>
<td>cumSum()</td>
<td>802</td>
</tr>
<tr>
<td>dataMat</td>
<td>805</td>
</tr>
<tr>
<td>diag()</td>
<td>810</td>
</tr>
<tr>
<td>dim()</td>
<td>810</td>
</tr>
<tr>
<td>dotP()</td>
<td>812</td>
</tr>
<tr>
<td>eigVc()</td>
<td>815</td>
</tr>
<tr>
<td>eigVl()</td>
<td>815</td>
</tr>
<tr>
<td>identity()</td>
<td>830</td>
</tr>
<tr>
<td>listMat()</td>
<td>837</td>
</tr>
<tr>
<td>LU</td>
<td>841</td>
</tr>
<tr>
<td>matData</td>
<td>841</td>
</tr>
<tr>
<td>matList()</td>
<td>842</td>
</tr>
<tr>
<td>max()</td>
<td>842</td>
</tr>
<tr>
<td>mean()</td>
<td>842</td>
</tr>
<tr>
<td>median()</td>
<td>842</td>
</tr>
<tr>
<td>min()</td>
<td>844</td>
</tr>
<tr>
<td>mRow()</td>
<td>844</td>
</tr>
<tr>
<td>mRowAdd()</td>
<td>844</td>
</tr>
<tr>
<td>newMat()</td>
<td>846</td>
</tr>
<tr>
<td>norm()</td>
<td>848</td>
</tr>
<tr>
<td>product()</td>
<td>855</td>
</tr>
<tr>
<td>QR</td>
<td>859</td>
</tr>
<tr>
<td>randMat()</td>
<td>862</td>
</tr>
<tr>
<td>ref()</td>
<td>863</td>
</tr>
<tr>
<td>rowAdd()</td>
<td>863</td>
</tr>
<tr>
<td>rowDim()</td>
<td>866</td>
</tr>
<tr>
<td>rowNorm()</td>
<td>866</td>
</tr>
<tr>
<td>rowSwap()</td>
<td>867</td>
</tr>
<tr>
<td>rref()</td>
<td>867</td>
</tr>
<tr>
<td>simlt()</td>
<td>876</td>
</tr>
<tr>
<td>stdDev()</td>
<td>882</td>
</tr>
<tr>
<td>sum()</td>
<td>884</td>
</tr>
<tr>
<td>stdDevPop()</td>
<td>882</td>
</tr>
<tr>
<td>T</td>
<td>885</td>
</tr>
<tr>
<td>unitV()</td>
<td>892</td>
</tr>
<tr>
<td>variance()</td>
<td>893</td>
</tr>
</tbody>
</table>

---

Appendix A: Functions and Instructions 783
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ans()</td>
<td>788</td>
</tr>
<tr>
<td>Archive</td>
<td>792</td>
</tr>
<tr>
<td>ClockOff</td>
<td>793</td>
</tr>
<tr>
<td>ClockOn</td>
<td>794</td>
</tr>
<tr>
<td>Custom</td>
<td>795</td>
</tr>
<tr>
<td>CustomOn</td>
<td>796</td>
</tr>
<tr>
<td>CustomOff</td>
<td>797</td>
</tr>
<tr>
<td>Define</td>
<td>798</td>
</tr>
<tr>
<td>DelVar</td>
<td>800</td>
</tr>
<tr>
<td>DelType</td>
<td>801</td>
</tr>
<tr>
<td>Disp</td>
<td>802</td>
</tr>
<tr>
<td>DispG</td>
<td>803</td>
</tr>
<tr>
<td>DropDown</td>
<td>804</td>
</tr>
<tr>
<td>else</td>
<td>805</td>
</tr>
<tr>
<td>EndCustom</td>
<td>806</td>
</tr>
<tr>
<td>EndFunc</td>
<td>807</td>
</tr>
<tr>
<td>EndFor</td>
<td>808</td>
</tr>
<tr>
<td>EndLoop</td>
<td>809</td>
</tr>
<tr>
<td>EndTry</td>
<td>810</td>
</tr>
<tr>
<td>EndWhile</td>
<td>811</td>
</tr>
<tr>
<td>Exec</td>
<td>812</td>
</tr>
<tr>
<td>Extern</td>
<td>813</td>
</tr>
<tr>
<td>Func</td>
<td>814</td>
</tr>
<tr>
<td>Get</td>
<td>815</td>
</tr>
<tr>
<td>getKey()</td>
<td>816</td>
</tr>
<tr>
<td>getMode()</td>
<td>817</td>
</tr>
<tr>
<td>Exit</td>
<td>818</td>
</tr>
<tr>
<td>if</td>
<td>819</td>
</tr>
<tr>
<td>Input</td>
<td>820</td>
</tr>
<tr>
<td>isCkOn()</td>
<td>821</td>
</tr>
<tr>
<td>isArch()</td>
<td>822</td>
</tr>
<tr>
<td>left()</td>
<td>823</td>
</tr>
<tr>
<td>Local</td>
<td>824</td>
</tr>
<tr>
<td>MoveVar</td>
<td>825</td>
</tr>
<tr>
<td>not</td>
<td>826</td>
</tr>
<tr>
<td>NewFold</td>
<td>827</td>
</tr>
<tr>
<td>NewProb</td>
<td>828</td>
</tr>
<tr>
<td>or</td>
<td>829</td>
</tr>
<tr>
<td>Output</td>
<td>830</td>
</tr>
<tr>
<td>part()</td>
<td>831</td>
</tr>
<tr>
<td>Item</td>
<td>832</td>
</tr>
<tr>
<td>IsCkOn()</td>
<td>833</td>
</tr>
<tr>
<td>IsLocked()</td>
<td>834</td>
</tr>
<tr>
<td>Lbl</td>
<td>835</td>
</tr>
<tr>
<td>InputStr</td>
<td>836</td>
</tr>
<tr>
<td>Output()</td>
<td>837</td>
</tr>
<tr>
<td>Prompt</td>
<td>838</td>
</tr>
<tr>
<td>Rename</td>
<td>839</td>
</tr>
<tr>
<td>Return</td>
<td>840</td>
</tr>
<tr>
<td>SendCalc</td>
<td>841</td>
</tr>
<tr>
<td>SendChat</td>
<td>842</td>
</tr>
<tr>
<td>Send</td>
<td>843</td>
</tr>
<tr>
<td>setFold()</td>
<td>844</td>
</tr>
<tr>
<td>setMode()</td>
<td>845</td>
</tr>
<tr>
<td>setTable()</td>
<td>846</td>
</tr>
<tr>
<td>setVar()</td>
<td>847</td>
</tr>
<tr>
<td>TimeConversion</td>
<td>848</td>
</tr>
<tr>
<td>setText()</td>
<td>849</td>
</tr>
<tr>
<td>Text</td>
<td>850</td>
</tr>
<tr>
<td>Toolbar</td>
<td>851</td>
</tr>
<tr>
<td>Unarchiv</td>
<td>852</td>
</tr>
<tr>
<td>Unlock</td>
<td>853</td>
</tr>
<tr>
<td>Unlock()</td>
<td>854</td>
</tr>
<tr>
<td>when()</td>
<td>855</td>
</tr>
<tr>
<td>While</td>
<td>856</td>
</tr>
<tr>
<td>xor</td>
<td>857</td>
</tr>
<tr>
<td>timeConvert</td>
<td>858</td>
</tr>
<tr>
<td>timeConversion</td>
<td>859</td>
</tr>
<tr>
<td>Try</td>
<td>860</td>
</tr>
<tr>
<td>then</td>
<td>861</td>
</tr>
<tr>
<td>Then</td>
<td>862</td>
</tr>
<tr>
<td>Title</td>
<td>863</td>
</tr>
<tr>
<td>Unlock()</td>
<td>864</td>
</tr>
<tr>
<td>Unlock</td>
<td>865</td>
</tr>
<tr>
<td>Unlock()</td>
<td>866</td>
</tr>
<tr>
<td>Unarchiv</td>
<td>867</td>
</tr>
<tr>
<td>Unlock()</td>
<td>868</td>
</tr>
<tr>
<td>Unlock()</td>
<td>869</td>
</tr>
<tr>
<td>Unlock()</td>
<td>870</td>
</tr>
<tr>
<td>unlock</td>
<td>871</td>
</tr>
<tr>
<td>Unarchiv</td>
<td>872</td>
</tr>
<tr>
<td>Unlock</td>
<td>873</td>
</tr>
<tr>
<td>Unlock()</td>
<td>874</td>
</tr>
<tr>
<td>Unlock()</td>
<td>875</td>
</tr>
<tr>
<td>Unlock()</td>
<td>876</td>
</tr>
<tr>
<td>Unlock()</td>
<td>877</td>
</tr>
<tr>
<td>Unlock()</td>
<td>878</td>
</tr>
<tr>
<td>Unlock()</td>
<td>879</td>
</tr>
<tr>
<td>Unlock()</td>
<td>880</td>
</tr>
<tr>
<td>Unlock()</td>
<td>881</td>
</tr>
<tr>
<td>Unlock()</td>
<td>882</td>
</tr>
<tr>
<td>Unlock()</td>
<td>883</td>
</tr>
<tr>
<td>Unlock()</td>
<td>884</td>
</tr>
<tr>
<td>Unlock()</td>
<td>885</td>
</tr>
<tr>
<td>Unlock()</td>
<td>886</td>
</tr>
<tr>
<td>Unlock()</td>
<td>887</td>
</tr>
<tr>
<td>Unlock()</td>
<td>888</td>
</tr>
<tr>
<td>Unlock()</td>
<td>889</td>
</tr>
<tr>
<td>Unlock()</td>
<td>890</td>
</tr>
<tr>
<td>Unlock()</td>
<td>891</td>
</tr>
<tr>
<td>Unlock()</td>
<td>892</td>
</tr>
<tr>
<td>Unlock()</td>
<td>893</td>
</tr>
<tr>
<td>Unlock()</td>
<td>894</td>
</tr>
</tbody>
</table>
### Statistics

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (factorial)</td>
<td>906</td>
</tr>
<tr>
<td>cumSum()</td>
<td>802</td>
</tr>
<tr>
<td>LnReg</td>
<td>838</td>
</tr>
<tr>
<td>median()</td>
<td>842</td>
</tr>
<tr>
<td>NewData</td>
<td>845</td>
</tr>
<tr>
<td>OneVar</td>
<td>849</td>
</tr>
<tr>
<td>PowerReg</td>
<td>855</td>
</tr>
<tr>
<td>rand()</td>
<td>861</td>
</tr>
<tr>
<td>ShowStat</td>
<td>875</td>
</tr>
<tr>
<td>SortD</td>
<td>881</td>
</tr>
<tr>
<td>TwoVar</td>
<td>892</td>
</tr>
<tr>
<td>cumSum()</td>
<td>819</td>
</tr>
<tr>
<td>ExpReg</td>
<td>819</td>
</tr>
<tr>
<td>Logistic</td>
<td>840</td>
</tr>
<tr>
<td>MedMed</td>
<td>843</td>
</tr>
<tr>
<td>NewPlot</td>
<td>847</td>
</tr>
<tr>
<td>PlotsOff</td>
<td>853</td>
</tr>
<tr>
<td>QuadReg</td>
<td>859</td>
</tr>
<tr>
<td>randNorm()</td>
<td>862</td>
</tr>
<tr>
<td>SinReg</td>
<td>878</td>
</tr>
<tr>
<td>stdDev()</td>
<td>882</td>
</tr>
<tr>
<td>stdDevPop()</td>
<td>882</td>
</tr>
<tr>
<td>CubicReg</td>
<td>801</td>
</tr>
<tr>
<td>LinReg</td>
<td>836</td>
</tr>
<tr>
<td>mean()</td>
<td>842</td>
</tr>
<tr>
<td>nCr()</td>
<td>845</td>
</tr>
<tr>
<td>nPr()</td>
<td>848</td>
</tr>
<tr>
<td>PlotsOn</td>
<td>853</td>
</tr>
<tr>
<td>QuartReg</td>
<td>860</td>
</tr>
<tr>
<td>RandSeed</td>
<td>862</td>
</tr>
<tr>
<td>SortA</td>
<td>881</td>
</tr>
<tr>
<td>variance()</td>
<td>893</td>
</tr>
</tbody>
</table>

### Strings

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp; (append)</td>
<td>907</td>
</tr>
<tr>
<td># (indirection)</td>
<td>908</td>
</tr>
<tr>
<td>char()</td>
<td>791</td>
</tr>
<tr>
<td>dim()</td>
<td>810</td>
</tr>
<tr>
<td>expr()</td>
<td>818</td>
</tr>
<tr>
<td>format()</td>
<td>822</td>
</tr>
<tr>
<td>inString()</td>
<td>832</td>
</tr>
<tr>
<td>left()</td>
<td>834</td>
</tr>
<tr>
<td>mid()</td>
<td>843</td>
</tr>
<tr>
<td>ord()</td>
<td>850</td>
</tr>
<tr>
<td>right()</td>
<td>865</td>
</tr>
<tr>
<td>rotate()</td>
<td>865</td>
</tr>
<tr>
<td>shift()</td>
<td>874</td>
</tr>
<tr>
<td>string()</td>
<td>883</td>
</tr>
</tbody>
</table>
Abs() MATH/Number menu

Abs(expression) =⇒ expression
Abs(list) =⇒ list
Abs(matrix) =⇒ matrix

Returns the absolute value of the argument.
If the argument is a complex number, returns the number's modulus.

Note: All undefined variables are treated as real variables.

Abs({p/2, e/3}) =⇒ {p/2, e/3}
Abs(2-3i) =⇒ √13
Abs(z) =⇒ |z|
Abs(x+y) =⇒ √(x²+y²)

And MATH/Test and MATH/Base menus

Boolean expression1 and expression2 =⇒ Boolean expression
Boolean list1 and list2 =⇒ Boolean list
Boolean matrix1 and matrix2 =⇒ Boolean matrix

Returns true or false or a simplified form of the original entry.

In Hex base mode:
0h7AC36 and 0h3D5F =⇒ 0h2C16
Important: Zero, not the letter O.

In Bin base mode:
0b100101 and 0b100 =⇒ 0b100

In Dec base mode:
37 and 0b100 =⇒ 4

Note: A binary entry can have up to 32 digits (not counting the 0b prefix). A hexadecimal entry can have up to 8 digits.
AndPic (picVar, row, column)

Displays the Graph screen and logically "ANDS" the picture stored in picVar and the current graph screen at pixel coordinates (row, column).

picVar must be a picture type.

Default coordinates are (0,0), which is the upper left corner of the screen.

In function graphing mode and Y= Editor:

\[ y_1(x) = \cos(x) \]
\[ [2] \text{ Style } = 3: \text{Square} \]
\[ [2] \text{ Zoom } = 7: \text{ZoomTrig} \]
\[ [2] = 2: \text{Save Copy As...} \]
Type = Picture, Variable = PIC1

\[ y_2(x) = \sin(x) \]
\[ [2] \text{ Style } = 3: \text{Square} \]
\[ [2] \text{ Zoom } = 7: \text{ZoomTrig} \]

angle() MATH/Complex menu

angle(expression) ⇒ expression

Returns the angle of expression, interpreting expression as a complex number.

Note: All undefined variables are treated as real variables.

In Degree angle mode:

angle(0+2i) \[\Rightarrow\] \[90] \n
In Gradian angle mode:

angle(0+3i) \[\Rightarrow\] \[100] \n
In Radian angle mode:

angle(1+i) \[\Rightarrow\] \[\frac{\pi}{4}] \n
angle(x+i y) \[\Rightarrow\]

\[
\text{angle(z)} = \frac{x \left( \text{sign}(z) - 1 \right)}{2} \]
\[
\text{angle}(x+i y) = \frac{x \text{sign}(y) - \text{tanh}^{-1} \left( \frac{y}{x} \right)}{2} \]

angle(list) ⇒ list
angle(matrix) ⇒ matrix

Returns a list or matrix of angles of the elements in list or matrix, interpreting each element as a complex number that represents a two-dimensional rectangular coordinate point.

In Radian angle mode:

angle({1+2i, 3+0i, 0+\frac{\pi}{4}i}) \[\Rightarrow\]

\[
\text{angle}(1+2i) = 1.0 \]
\[
\text{angle}(3+0i) = 0.0 \]
\[
\text{angle}(0+\frac{\pi}{4}i) = 0.2 \]
### ans()  
**[ANS] key**

<table>
<thead>
<tr>
<th>Function</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ans()</td>
<td>⇒⇒⇒ value</td>
<td>To use ans() to generate the Fibonacci sequence on the Home screen, press:</td>
</tr>
<tr>
<td>ans(integer)</td>
<td>⇒⇒⇒ value</td>
<td>Returns a previous answer from the Home screen history area.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>integer, if included, specifies which previous answer to recall. Valid range for integer is from 1 to 99 and cannot be an expression. Default is 1, the most recent answer.</td>
</tr>
</tbody>
</table>

### approx()  
**MATH/Algebra menu**

<table>
<thead>
<tr>
<th>Function</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>approx(expression)</td>
<td>⇒⇒⇒ value</td>
<td>Returns the evaluation of expression as a decimal value, when possible, regardless of the current Exact/Approx mode. This is equivalent to entering expression and pressing <code>=</code> on the Home screen.</td>
</tr>
</tbody>
</table>

### Archive  
**CATALOG**

<table>
<thead>
<tr>
<th>Function</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive</td>
<td>var1, var2, ...</td>
<td>Moves the specified variables from RAM to the user data archive memory. You can access an archived variable the same as you would a variable in RAM. However, you cannot delete, rename, or store to an archived variable because it is locked automatically. To unarchive variables, use Unarchiv.</td>
</tr>
</tbody>
</table>

### arcLen()  
**MATH/Calculus menu**

<table>
<thead>
<tr>
<th>Function</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arcLen(expression1, var, start, end)</td>
<td>⇒⇒⇒ expression</td>
<td>Returns the arc length of expression1 from start to end with respect to variable var. Regardless of the graphing mode, arc length is calculated as an integral assuming a function mode definition.</td>
</tr>
<tr>
<td>arcLen(list1, var, start, end)</td>
<td>⇒⇒⇒ list</td>
<td>Returns a list of the arc lengths of each element of list1 from start to end with respect to var.</td>
</tr>
</tbody>
</table>
**augment() MATH/Matrix menu**

\[
\text{augment}([1, 2, 3], [4, 5, 6]) \Rightarrow [1, 2, 3, 4, 5, 6]
\]

**augment(list1, list2) ⇒ list**

Returns a new list that is list2 appended to the end of list1.

\[
\text{augment}([1, 2, 3], [4, 5, 6]) \Rightarrow [1, 2, 3, 4, 5, 6]
\]

**augment(matrix1, matrix2) ⇒ matrix**

Returns a new matrix that is matrix2 appended to matrix1. When the "," character is used, the matrices must have equal row dimensions, and matrix2 is appended to matrix1 as new columns. When the ";" character is used, the matrices must have equal column dimensions, and matrix2 is appended to matrix1 as new rows. Does not alter matrix1 or matrix2.

\[
\begin{bmatrix}
1 & 2 \\
3 & 4
\end{bmatrix}
\]

\[
\begin{bmatrix}
5 & 6
\end{bmatrix}
\]

**avgRC() CATALOG**

\[
\text{avgRC(expression1, var, h)} \Rightarrow \text{expression}
\]

Returns the forward-difference quotient (average rate of change).

\[
\text{avgRC}(x^2, x, .1) \Rightarrow 2 \cdot (x + .1)
\]

**avgRC(expression1, var, h) ⇒ expression**

Returns the forward-difference quotient (average rate of change).

\[
\text{avgRC}(\sin(x), x, .1) \Rightarrow \sin(x + .1) - \sin(x)
\]

**avgRC(expression1, var, h) ⇒ expression**

Returns the forward-difference quotient (average rate of change).

\[
\text{avgRC}(\sin(x), x, .1) \Rightarrow \sin(x + .1) - \sin(x)
\]

**avgRC(expression1, var, h) ⇒ expression**

Returns the forward-difference quotient (average rate of change).

\[
\text{avgRC}(\sin(x), x, .1) \Rightarrow \sin(x + .1) - \sin(x)
\]

**Bin MATH/Base menu**

\[
\text{Bin} \Rightarrow \text{integer}
\]

Converts integer to a binary number. Binary or hexadecimal numbers always have a 0b or 0h prefix, respectively.

- Zero, not the letter O, followed by b or h.
- 0b binaryNumber
- 0h hexadecimalNumber

A binary number can have up to 32 digits. A hexadecimal number can have up to 8.

Without a prefix, integer is treated as decimal (base 10). The result is displayed in binary, regardless of the Base mode.

If you enter a decimal integer that is too large for a signed, 32-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range.
BldData CATALOG

BldData [dataVar]

Creates data variable dataVar based on the information used to plot the current graph.

BldData is valid in all graphing modes.

If dataVar is omitted, the data is stored in the system variable sysData.

Note: The first time you start the Data/Matrix Editor after using BldData, dataVar or sysData (depending on the argument you used with BldData) is set as the current data variable.

The incremental values used for any independent variables (x in the example to the right) are calculated according to the Window variable values.

For information about the increments used to evaluate a graph, refer to the module that describes that graphing mode.

3D graphing mode has two independent variables. In the sample data to the right, notice that x remains constant as y increments through its range of values.

Then, x increments to its next value and y again increments through its range. This pattern continues until x has incremented through its range.

Note: The following sample data is from a 3D graph.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>200</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>300</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>400</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>500</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

In function graphing mode and Radian angle mode:

\[8 \sin(x) \Rightarrow y_1(x) \text{ ENTER } \text{ Done}\]

\[2 \sin(x) \Rightarrow y_2(x) \text{ ENTER } \text{ Done}\]

ZoomStd \text{ ENTER}\n
\[
\begin{align*}
t_1 \Rightarrow \text{Done} \\
8 \sin(x) \Rightarrow y_1(x) \\
2 \sin(x) \Rightarrow y_2(x)
\end{align*}
\]

ceiling() MATH/Number menu

ceiling(expression) ⇒ integer

Returns the nearest integer that is ≤ the argument.

The argument can be a real or a complex number.

Note: See also floor().

ceiling(list) ⇒ list

ceiling(matrix) ⇒ matrix

Returns a list or matrix of the ceiling of each element.
Appendix A: Functions and Instructions

\textbf{cFactor()} \quad \text{MATH/Algebra/Complex menu}

\texttt{cFactor(expression1, var)} \Rightarrow expression
\texttt{cFactor(list, var)} \Rightarrow list
\texttt{cFactor(matrix, var)} \Rightarrow matrix

\texttt{cFactor(expression1, var)} \Rightarrow expression
\texttt{cFactor(list, var)} \Rightarrow list
\texttt{cFactor(matrix, var)} \Rightarrow matrix

\texttt{cFactor(expression1)} \Rightarrow expression1 factored with respect to all of its variables over a common denominator.

\texttt{cFactor(expr)} is factored as much as possible toward linear rational factors even if this introduces new non-real numbers. This alternative is appropriate if you want factorization with respect to more than one variable.

\texttt{cFactor(a^3 - x^2 + a, x)} \Rightarrow a \cdot (a + \sqrt{x}) \cdot (a - \sqrt{x}) \cdot (x + \sqrt{x}) \cdot (x - \sqrt{x})

\texttt{cFactor(x^2 + 4/9)} \Rightarrow (\sqrt{3} \cdot x + \sqrt[3]{2} \cdot \sqrt{i}) \cdot \sqrt{3} \cdot x - \sqrt[3]{2} \cdot \sqrt{i}

\texttt{cFactor(x^2 + 3)} \Rightarrow \sqrt{x} + 3

\texttt{cFactor(x^2 + a)} \Rightarrow \sqrt{x} + a

\texttt{cFactor(a^3 - x^2 + a, x)} \Rightarrow a \cdot (a + 1) \cdot (x + \sqrt{3} \cdot \sqrt{i}) \cdot (x - \sqrt{3} \cdot \sqrt{i})

\texttt{cFactor(x^2 + 3, x)} \Rightarrow (x + \sqrt{3}) \cdot (x - \sqrt{3})

\texttt{cFactor(x^2 + a, x)} \Rightarrow (x + \sqrt{a} \cdot \sqrt{i}) \cdot (x - \sqrt{a} \cdot \sqrt{i})

For the \texttt{AUTO} setting of the Exact/Approx mode, including \texttt{var} also permits approximation with floating-point coefficients where irrational coefficients cannot be explicitly expressed concisely in terms of the built-in functions. Even when there is only one variable, including \texttt{var} might yield more complete factorization.

\textbf{char()} \quad \text{MATH/String menu}

\texttt{char(integer)} \Rightarrow character

Returns a character string containing the character numbered \texttt{integer} from the TI-89 Titanium/Voyage™ 200 character set. See Appendix B for a complete listing of character codes. The valid range for \texttt{integer} is 0–255.

\texttt{char(38)} \Rightarrow "&"

\texttt{char(65)} \Rightarrow "A"

\textbf{checkTmr()} \quad \text{CATALOG}

\texttt{checkTmr(starttime)} \Rightarrow integer

Returns an integer representing the number of seconds that have elapsed since a timer was started. \texttt{starttime} is an integer returned from the \texttt{startTmr()} function.

You can also use a list or matrix of \texttt{starttime} integers. Valid \texttt{starttime} integers must fall between 0 and the current time of the clock. You can run multiple timers simultaneously.

\textbf{Note:} See also \texttt{startTmr()} and \texttt{timeCnv()}.
Circle | CATALOG
---|---
Circle \( x, y, r, \) drawMode\)

- Draws a circle with its center at window coordinates \((x, y)\) and with a radius of \(r\).
- \(x, y,\) and \(r\) must be real values.
- If drawMode = 1, draws the circle (default).
- If drawMode = 0, turns off the circle.
- If drawMode = -1, inverts pixels along the circle.

**Note:** Regraphing erases all drawn items. See also \(\text{PxlCrcl}\).

ClockOff | CATALOG
---|---
ClockOff

- Turns the clock OFF.

ClockOn | CATALOG
---|---
ClockOn

- Turns the clock ON.

ClrDraw | CATALOG
---|---
ClrDraw

- Clears the Graph screen and resets the Smart Graph feature so that the next time the Graph screen is displayed, the graph will be redrawn.
- While viewing the Graph screen, you can clear all drawn items (such as lines and points) by pressing \(\text{F6}\) (ReGraph) or pressing \(\text{2} \) and selecting 1:ClrDraw.

ClrErr | CATALOG
---|---
ClrErr

- Clears the error status. It sets errornum to zero and clears the internal error context variables.
- The **Else** clause of the **Try...EndTry** in the program should use **ClrErr** or **PassErr**. If the error is to be processed or ignored, use **ClrErr**. If what to do with the error is not known, use **PassErr** to send it to the next error handler. If there are no more pending **Try...EndTry** error handlers, the error dialog box will be displayed as normal.

**Note:** See also **PassErr** and **Try**.

Program listing:
```
:clearerr()
:prgm
:plotsOff:fnOff:zoomStd
:for 1,0,238
:  @x=xmin+i\(\times\)xord
:  try
:    ptOn xord,ln(xord)
:  else
:    if errornum=800 or errornum=260 then
:      clrErr ; clear the error
:    else
:      passErr ; pass on any other error
:    endif
:    endif
:  endif
:  try
:endFor
:endprgm
```
ClrGraph  CATALOG

ClrGraph

Clears any functions or expressions that were graphed with the Graph command or were created with the Table command. (See Graph or Table.)

Any previously selected Y= functions will be graphed the next time that the graph is displayed.

ClrHome  CATALOG

ClrHome

Clears all items stored in the entry() and ans() Home screen history area. Does not clear the current entry line.

While viewing the Home screen, you can clear the history area by pressing [F1] and selecting 8:Clear Home.

For functions such as solve() that return arbitrary constants or integers (@1, @2, etc.), ClrHome resets the suffix to 1.

ClrIO  CATALOG

ClrIO

Clears the Program I/O screen.

ClrTable  CATALOG

ClrTable

Clears all table values. Applies only to the ASK setting on the Table Setup dialog box.

While viewing the Table screen in Ask mode, you can clear the values by pressing [F1] and selecting 8:Clear Table.

colDim()  MATH/Matrix/Dimensions menu

\[ \text{colDim(matrix)} \Rightarrow \text{expression} \]

Returns the number of columns contained in matrix.

Note: See also rowDim().

colNorm()  MATH/Matrix/Norms menu

\[ \text{colNorm(matrix)} \Rightarrow \text{expression} \]

Returns the maximum of the sums of the absolute values of the elements in the columns in matrix.

Note: Undefined matrix elements are not allowed. See also rowNorm().

comDenom()  MATH/Algebra menu

\[ \text{comDenom(expression[, var])} \Rightarrow \text{expression} \]

\[ \text{comDenom(list[, var])} \Rightarrow \text{list} \]

\[ \text{comDenom(matrix[, var])} \Rightarrow \text{matrix} \]

\[ \text{comDenom(expression)} \text{ returns a reduced ratio of a fully expanded numerator over a fully expanded denominator.} \]
comDenom(expression1, var) returns a reduced ratio of numerator and denominator expanded with respect to var. The terms and their factors are sorted with var as the main variable. Similar powers of var are collected. There might be some incidental factoring of the collected coefficients. Compared to omitting var, this often saves time, memory, and screen space, while making the expression more comprehensible. It also makes subsequent operations on the result faster and less likely to exhaust memory.

\[
\text{comDenom}\left(\frac{y^2+y}{(x+1)^2+y^2+y}, x\right)
\]

If var does not occur in expression1, comDenom(expression1, var) returns a reduced ratio of an unexpanded numerator over an unexpanded denominator. Such results usually save even more time, memory, and screen space. Such partially factored results also make subsequent operations on the result much faster and much less likely to exhaust memory.

Even when there is no denominator, the comden function is often a fast way to achieve partial factorization if factor() is too slow or if it exhausts memory.

**Hint:** Enter this comden() function definition and routinely try it as an alternative to comDenom() and factor().

**conj()**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conj(expression)</td>
<td>Returns the complex conjugate of the argument.</td>
</tr>
</tbody>
</table>

**Note:** All undefined variables +are treated as real variables.

**CopyVar**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyVar var1, var2</td>
<td>Copies the contents of variable var1 to var2. If var2 does not exist, CopyVar creates it.</td>
</tr>
</tbody>
</table>

**Note:** CopyVar is similar to the store instruction (\(\leftarrow\)) when you are copying an expression, list, matrix, or character string except that no simplification takes place when using CopyVar. You must use CopyVar with non-algebraic variable types such as Pic and GDB variables.
### \( \text{cos()} \) Function

**Keys:**
- \( \text{cos(expression)} \Rightarrow \) expression
- \( \text{cos(list)} \Rightarrow \) list

- \( \text{cos(expression)} \) returns the cosine of the argument as an expression.
- \( \text{cos(list)} \) returns a list of the cosines of all elements in list.

**Note:** The argument is interpreted as a degree, gradian or radian angle, according to the current angle mode setting. You can use \( \text{deg} \), \( \text{grad} \) or \( \text{rad} \) to override the angle mode temporarily.

**Example:**

In Degree angle mode:
- \( \text{cos}((\pi/4) \text{ rad}) \) \( \to \) \( \sqrt{2}/2 \)
- \( \text{cos}(45) \) \( \to \) \( \sqrt{2}/2 \)
- \( \text{cos}([0, 60, 90]) \) \( \to \) \( \{1, 1/2, 0\} \)

In Gradian angle mode:
- \( \text{cos}([0, 50, 100]) \) \( \to \) \( \{1, \sqrt{2}/2, 0\} \)

In Radian angle mode:
- \( \text{cos}(\pi/4) \) \( \to \) \( \sqrt{2}/2 \)
- \( \text{cos}(45) \) \( \to \) \( \sqrt{2}/2 \)

#### \( \text{cos(squareMatrix)} \) Function

Returns the matrix cosine of \( \text{squareMatrix} \). This is not the same as calculating the cosine of each element.

When a scalar function \( f(A) \) operates on \( \text{squareMatrix} \), the result is calculated by the algorithm:

1. Compute the eigenvalues (\( \lambda_i \)) and eigenvectors (\( \mathbf{V}_i \)) of \( A \).

   \( \text{squareMatrix} \) must be diagonalizable. Also, it cannot have symbolic variables that have not been assigned a value.

2. Form the matrices:

   \[
   \mathbf{B} = \begin{bmatrix}
   \lambda_1 & 0 & \cdots & 0 \\
   0 & \lambda_2 & \cdots & 0 \\
   0 & 0 & \cdots & 0 \\
   0 & 0 & \cdots & \lambda_n
   \end{bmatrix}
   \]

   and \( \mathbf{X} = [\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_n] \)

3. Then \( A = \mathbf{X} \mathbf{B} \mathbf{X}^{-1} \) and \( f(A) = \mathbf{X} f(\mathbf{B}) \mathbf{X}^{-1} \). For example, \( \text{cos}(A) = \mathbf{X} \text{cos}(\mathbf{B}) \mathbf{X}^{-1} \) where:

   \[
   \text{cos} \mathbf{B} = \begin{bmatrix}
   \cos(\lambda_1) & 0 & \cdots & 0 \\
   0 & \cos(\lambda_2) & \cdots & 0 \\
   0 & 0 & \cdots & \cos(\lambda_n) \\
   0 & 0 & \cdots & \cos(\lambda_n)
   \end{bmatrix}
   \]

All computations are performed using floating-point arithmetic.

### Appendix A: Functions and Instructions

795


\[ \cos^{-1}(\theta) \]

- **key**
- **\( \cos^{-1}(\text{expression}) \Rightarrow \text{expression} \)**
- **\( \cos^{-1}(\text{list}) \Rightarrow \text{list} \)**

**\( \cos^{-1}(\text{expression}) \)** returns the angle whose cosine is \( \text{expression} \) as an expression.

**\( \cos^{-1}(\text{list}) \)** returns a list of the inverse cosines of each element of \( \text{list} \).

**Note:** The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

**In Degree angle mode:**

\[ \cos^{-1}(1) \]

**In Gradian angle mode:**

\[ \cos^{-1}(0) \]

**In Radian angle mode:**

\[ \cos^{-1}\{0.2, 0.5\} \]

\[ \frac{\pi}{6} \quad 1.369... \quad 1.047... \]

\[ \cos^{-1}(\text{squareMatrix}) \Rightarrow \text{squareMatrix} \]

**MATH/Hyperbolic menu**

- **cosh()**
- **\( \cosh(\text{expression}) \Rightarrow \text{expression} \)**
- **\( \cosh(\text{list}) \Rightarrow \text{list} \)**

**cosh()** returns the hyperbolic cosine of the argument as an expression.

**cosh()** returns a list of the hyperbolic cosines of each element of \( \text{list} \).

**In Radian angle mode:**

\[ \cosh(1.2) \]

\[ 1.810... \]

**cosh()** returns the matrix of the hyperbolic cosines of each element of \( \text{list} \).

**In Radian angle mode and Rectangular complex format mode:**

\[ \cosh([1.5, 3.4, 2.1; 6, -2.1]) \]

\[ \begin{bmatrix} 1.734...+0.646...i & -1.490...+2.105...i \\ -0.725...+1.515...i & 0.623...+1.778...i \\ -2.083...+2.632...i & 1.790...-1.271...i \end{bmatrix} \]

\[ \cosh^{-1}(\text{squareMatrix}) \Rightarrow \text{squareMatrix} \]

**MATH/Hyperbolic menu**

- **cosh^{-1}(\text{expression}) \Rightarrow \text{expression} \)**
- **\( \cosh^{-1}(\text{list}) \Rightarrow \text{list} \)**

**cosh^{-1}(\text{expression})** returns the inverse hyperbolic cosine of the argument as an expression.

**cosh^{-1}(\text{list})** returns a list of the inverse hyperbolic cosines of each element of \( \text{list} \).

**In Radian angle mode:**

\[ \cosh^{-1}(1) \]

\[ 0 \]

**In Radian angle mode and Rectangular complex format mode:**

\[ \cosh^{-1}([1.5, 3.4, 2.1; 6, -2.1]) \]

\[ \begin{bmatrix} 2.525...+1.734...i & -0.009...-1.490...i \\ -0.486...-0.725...i & 1.662...+0.623...i \\ -3.322...-2.083...i & 1.267...+1.790...i \end{bmatrix} \]
Appendix A: Functions and Instructions

**cot()** MATH/Trig menu

cot(expression1) ⇒ expression

cot(list1) ⇒ list

Returns the cotangent of expression1 or returns a list of the cotangents of all elements in list1.

**Note:** The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

In Degree angle mode:

cot(45) [ENTER] 1

cot(50) [ENTER] 1

cot((1,2,1,3)) [ENTER] \( \frac{1}{\tan(1)} \cdot \frac{1}{\tan(2)} \)

In Gradian angle mode:

cot(50) [ENTER] 1

In Radian angle mode:

cot((1,2,1,3)) [ENTER] \( \frac{1}{\tan(1)} \cdot \frac{1}{\tan(2)} \)

**cot^−1()** MATH/Trig menu

cot^−1(expression1) ⇒ expression

cot^−1(list1) ⇒ list

Returns the angle whose cotangent is expression1 or returns a list of the inverse cotangents of each element of list1.

**Note:** The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

In Degree angle mode:

cot^−1(1) [ENTER] 45

cot^−1(1) [ENTER] 50

cot^−1(1) [ENTER] \( \frac{x}{4} \)

**coth()** MATH/Hyperbolic menu

coth(expression1) ⇒ expression

coth(list1) ⇒ list

Returns the hyperbolic cotangent of expression1 or returns a list of the hyperbolic cotangents of all elements of list1.

coth(1.2) [ENTER] 1.199...

coth(1.3.2) [ENTER] \( \frac{1}{\tanh(1)} \cdot \frac{1}{\tanh(2)} \)

**coth^−1()** MATH/Hyperbolic menu

coth^−1(expression1) ⇒ expression

coth^−1(list1) ⇒ list

Returns the inverse hyperbolic cotangent of expression1 or returns a list containing the inverse hyperbolic cotangents of each element of list1.

coth^−1(3.5) [ENTER] .293...

coth^−1(-2.2.1.6) [ENTER] \( \frac{-\ln(3)}{2} \cdot \frac{.518 \cdot \ln(7/5)}{2} \)

**crossP()** MATH/Matrix/Vector ops menu

crossP(list1, list2) ⇒ list

crossP(vector1, vector2) ⇒ vector

Returns the cross product of list1 and list2 as a list.

list1 and list2 must have equal dimension, and the dimension must be either 2 or 3.

Returns a row or column vector (depending on the arguments) that is the cross product of vector1 and vector2.

Both vector1 and vector2 must be row vectors, or both must be column vectors. Both vectors must have equal dimension, and the dimension must be either 2 or 3.
**csc()**  
MATH/Trig menu

- `csc(expression)` → `expression`
- `csc(list)` → `list`

Returns the cosecant of `expression` or returns a list containing the cosecants of all elements in `list`.

In Degree mode:
- `csc(45)` ENTER \( \sqrt{2} \)

In Gradian mode:
- `csc(50)` ENTER \( \sqrt{2} \)

In Radian mode:
- `csc(\pi/2, \pi/3)` ENTER \[ \{1, 2, 3\} \]

---

**csc⁻¹()**  
MATH/Trig menu

- `csc⁻¹(expression)` → `expression`
- `csc⁻¹(list)` → `list`

Returns the angle whose cosecant is `expression` or returns a list containing the inverse cosecants of each element of `list`.

**Note:** The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

In Degree mode:
- `csc⁻¹(1)` ENTER \( 90 \)

In Gradian mode:
- `csc⁻¹(1)` ENTER \( 100 \)

In Radian mode:
- `csc⁻¹(1, 4, 6)` ENTER \[ \{ \pi/2, \pi, \pi/6 \} \]

---

**csch()**  
MATH/Hyperbolic menu

- `csch(expression)` → `expression`
- `csch(list)` → `list`

Returns the hyperbolic cosecant of `expression` or returns a list of the hyperbolic cosecants of all elements of `list`.

- `csch(3)` ENTER \( \frac{1}{\sinh(3)} \)

- `csch(1, 2, 4)` ENTER \[ \{ \frac{1}{\sinh(1)}, \frac{1}{\sinh(2)}, \frac{1}{\sinh(4)} \} \]

---

**csch⁻¹()**  
MATH/Hyperbolic menu

- `csch⁻¹(expression)` → `expression`
- `csch⁻¹(list)` → `list`

Returns the inverse hyperbolic cosecant of `expression` or returns a list containing the inverse hyperbolic cosecants of each element of `list`.

- `csch⁻¹(1)` ENTER \( \sinh^{-1}(1) \)

- `csch⁻¹(1, 2, 3)` ENTER \[ \{ \sinh^{-1}(1), \sinh^{-1}(2), \sinh^{-1}(3) \} \]

---

**cSolve()**  
MATH/Algebra/Complex menu

- `cSolve(equation, var)` → `Boolean expression`

Returns candidate complex solutions of an equation for `var`. The goal is to produce candidates for all real and non-real solutions. Even if the equation is real, `cSolve()` allows non-real results in real mode.

Although the TI-89 Titanium/Voyage™ 200 processes all undefined variables that do not end with an underscore (_), as if they were real, `cSolve()` can solve polynomial equations for complex solutions.

- `cSolve(x^3 = -1, x)` ENTER
- `solve(x^3 = -1, x)` ENTER
**Notes**

- **cSolve()**: temporarily sets the domain to complex during the solution even if the current domain is real. In the complex domain, fractional powers having odd denominators use the principal rather than the real branch. Consequently, solutions from `solve()` to equations involving such fractional powers are not necessarily a subset of those from `cSolve()`.

- **cSolve()** starts with exact symbolic methods. Except in EXACT mode, `cSolve()` also uses iterative approximate complex polynomial factoring, if necessary.

- **Note**: See also `cZeros()`, `solve()`, and `zeros()`.

- **Note**: If equation is non-polynomial with functions such as `abs()`, `angle()`, `conj()`, `real()`, or `imag()`, you should place an underscore (`_`) at the end of var. By default, a variable is treated as a real value.

- **Note**: If all of the equations are polynomials and if you do NOT specify any initial guesses, `cSolve()` uses the lexical Gröbner/Buchberger elimination method to attempt to determine all complex solutions.

- Complex solutions can include both real and non-real solutions, as in the example to the right.
Simultaneous polynomial equations can have extra variables that have no values, but represent given numeric values that could be substituted later.

\[
\begin{align*}
cSolve(u_\cdot v_\cdot u_\cdot c_\cdot v_\cdot \text{and} & \quad v_\cdot 2\cdot u_\cdot {\{u_\cdot v_\}}) & \text{(ENTER)} \\
& \quad u_\cdot \frac{-1\sqrt{-1\cdot c_\cdot -1}}{2} \text{and } v_\cdot \frac{1\sqrt{-1\cdot c_\cdot -1}}{2} \\
or & \quad u_\cdot \text{and } v_\cdot \frac{-1\sqrt{-1\cdot c_\cdot -1}}{2} \\
or & \quad u_\cdot = 0 \text{ and } v_\cdot = 0
\end{align*}
\]

You can also include solution variables that do not appear in the equations. These solutions show how families of solutions might contain arbitrary constants of the form \( @k \), where \( k \) is an integer suffix from 1 through 255. The suffix resets to 1 when you use \texttt{ClrHome} or \texttt{\&8:Clear Home}.

For polynomial systems, computation time or memory exhaustion may depend strongly on the order in which you list solution variables. If your initial choice exhausts memory or your patience, try rearranging the variables in the equations and/or \texttt{varOrGuess} list.

If you do not include any guesses and if any equation is non-polynomial in any variable but all equations are linear in all solution variables, \texttt{cSolve()} uses Gaussian elimination to attempt to determine all solutions.

If a system is neither polynomial in all of its variables nor linear in its solution variables, \texttt{cSolve()} determines at most one solution using an approximate iterative method. To do so, the number of solution variables must equal the number of equations, and all other variables in the equations must simplify to numbers.

A non-real guess is often necessary to determine a non-real solution. For convergence, a guess might have to be rather close to a solution.
CubicReg  MATH/Statistics/Regressions menu

CubicReg list1, list2, list3, list4, list5

Calculates the cubic polynomial regression and updates all the statistics variables.

All the lists must have equal dimensions except for list5.

list1 represents xlist.
list2 represents ylist.
list3 represents frequency.
list4 represents category codes.
list5 represents category include list.

Note: list1 through list4 must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). list5 does not have to be a variable name and cannot be c1–c99.

In function graphing mode.

{0,1,2,3} → L1 ENTER
{0,2,3,4} → L2 ENTER
CubicReg L1,L2 ENTER Done
ShowStat ENTER

regeq(x) → y1(x) ENTER Done
NewPlot 1,1,L1,L2 ENTER Done

Appendix A: Functions and Instructions  801
### cumSum() MATH/List menu

<table>
<thead>
<tr>
<th><strong>cumSum(list)</strong> ⇒ <strong>list</strong></th>
<th><strong>cumSum({1,2,3,4})</strong> ENTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a list of the cumulative sums of the elements in list, starting at element 1.</td>
<td>{1 3 6 10}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>cumSum(matrix)</strong> ⇒ <strong>matrix</strong></th>
<th><strong>cumSum([1,2;3,4])</strong> ENTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a matrix of the cumulative sums of the elements in matrix. Each element is the cumulative sum of the column from top to bottom.</td>
<td>![Matrix Cumulative Sums]</td>
</tr>
</tbody>
</table>

### CustmOff CATALOG

**CustmOff**

Removes a custom toolbar.

*CustmOn* and *CustmOff* enable a program to control a custom toolbar. Manually, you can press ![CUSTOM] to toggle a custom toolbar on and off. Also, a custom toolbar is removed automatically when you change applications.

### CustmOn CATALOG

**CustmOn**

Activates a custom toolbar that has already been set up in a *Custom...EndCustm* block.

*CustmOn* and *CustmOff* enable a program to control a custom toolbar. Manually, you can press ![CUSTOM] to toggle a custom toolbar on and off.

### Custom ![CUSTOM] key

**Custom** block

Sets up a toolbar that is activated when you press ![CUSTOM]. It is very similar to the *ToolBar* instruction except that *Title* and *Item* statements cannot have labels.

*block* can be either a single statement or a series of statements separated with the `:` character.

**Note:** `[CUSTOM]` acts as a toggle. The first instance invokes the menu, and the second instance removes the menu. The menu is removed also when you change applications.

---

**Program listing:**

```
:Prgm
:Custom

:Title "Lists"
:Item "List1"
:Item "Scores"

:Title "Fractions"
:Item "f(x)"
:Item "h(x)"

:Title "Graph"
:EndCustm
:EndPrgm
```
**Catalog**

**Cycle**

Transfers program control immediately to the next iteration of the current loop (For, While, or Loop).

**Cycle** is not allowed outside the three looping structures (For, While, or Loop).

Program listing:

```plaintext
:temp
:For i,1,100,1
:If i=50
:Cycle
:temp+i
:EndFor
:Disp temp
```

Contents of `temp` after execution: 5000

**Catalog**

**CyclePic**

**CyclePic** `picNameString`, `n [, [wait] , [cycles], [direction]]`

Displays all the PIC variables specified and at the specified interval. The user has optional control over the time between pictures, the number of times to cycle through the pictures, and the direction to go, circular or forward and backwards.

`direction` is 1 for circular or \( \pi \) for forward and backwards. Default = 1.

1. Save three pics named `pic1`, `pic2`, and `pic3`.
2. Enter: `CyclePic "pic",3,.5,4, ë`
3. The three pictures (3) will be displayed automatically—one-half second (0.5) between pictures, for four cycles (4), and forward and backwards (ë).

**Catalog**

**Cylind**

**Cylind** displays the row or column vector in cylindrical form \([r \ \angle \ \theta, z]\).

`vector` must have exactly three elements. It can be either a row or a column.

```plaintext
[2,2,3] \rightarrow \text{Cylind}
\[2\sqrt{2} \ \frac{\pi}{4} \ \ 3\]
```

**Catalog**

**cZeros()**

**cZeros()** `expression`, `var` → `list`

Returns a list of candidate real and non-real values of `var` that make `expression=0`. `cZeros()` does this by computing `exp(list(cSolve(expression=0,var),var))`. Otherwise, `cZeros()` is similar to `zeros()`.

**Note:** See also `cSolve()`, `solve()`, and `zeros()`.

**Note:** If `expression` is non-polynomial with functions such as `abs()`, `angle()`, `conj()`, `real()`, or `imag()`, you should place an underscore (\( _{} \)) at the end of `var`. By default, a variable is treated as a real value. If you use `var_`, the variable is treated as complex.

You should also use `var_` for any other variables in `expression` that might have unreal values. Otherwise, you may receive unexpected results.

```plaintext
cZeros(x^5+4x^4+5x^3-i6x^2-3x,x) \rightarrow \{1+i, 1.073*\pi\}
```

**cZeros()** `expression1, expression2[, ... ]]`, `{varOrGuess1, varOrGuess2[, ... ]]` → `matrix`

Returns candidate positions where the expressions are zero simultaneously. Each `varOrGuess` specifies an unknown whose value you seek.

```plaintext
cZeros(expression1, expression2[, ... ]) \rightarrow \text{matrix}
```
Optionally, you can specify an initial guess for a variable. Each varOrGuess must have the form:

variable
– or –
variable = real or non-real number

For example, \( x \) is valid and so is \( x = 3 + i \).

If all of the expressions are polynomials and you do NOT specify any initial guesses, \( \text{cZeros()} \) uses the lexical Gröbner/Buchberger elimination method to attempt to determine all complex zeros.

Complex zeros can include both real and non-real zeros, as in the example to the right.

Each row of the resulting matrix represents an alternate zero, with the components ordered the same as the varOrGuess list. To extract a row, index the matrix by \([\text{row}]\).

You can also include unknown variables that do not appear in the expressions. These zeros show how families of zeros might contain arbitrary constants of the form \( @k \), where \( k \) is an integer suffix from 1 through 255. The suffix resets to 1 when you use \( \text{ClrHome} \) or \( \text{ƒ} 8: \text{Clear Home} \).

For polynomial systems, computation time or memory exhaustion may depend strongly on the order in which you list unknowns. If your initial choice exhausts memory or your patience, try rearranging the variables in the expressions and/or varOrGuess list.

If you do not include any guesses and if any expression is non-polynomial in any variable but all expressions are linear in all unknowns, \( \text{cZeros()} \) uses Gaussian elimination to attempt to determine all zeros.

If a system is neither polynomial in all of its variables nor linear in its unknowns, \( \text{cZeros()} \) determines at most one zero using an approximate iterative method. To do so, the number of unknowns must equal the number of expressions, and all other variables in the expressions must simplify to numbers.

Note: The following examples use an underscore \( _{\text{\textcopyright}} \) so that the variables will be treated as complex.
A non-real guess is often necessary to determine a non-real zero. For convergence, a guess might have to be rather close to a zero.

\[
c\text{Zeros}\left\{e^{z_1}z - z_2\right\}.
\{z_1 = 1 + i\} \quad \text{ENTER}
\{1.49 + 4.80i, 1.588 + 1.540i\}
\]

\(d()\)  key or MATH/Calculus menu

\[d(\text{expression1}, \text{var}, \text{order}) \Rightarrow \text{expression}\]
\[d(\text{list1}, \text{var}, \text{order}) \Rightarrow \text{list}\]
\[d(\text{matrix1}, \text{var}, \text{order}) \Rightarrow \text{matrix}\]

Returns the first derivative of \(\text{expression1}\) with respect to variable \(\text{var}\). \(\text{expression1}\) can be a list or a matrix.

\(\text{order}\), if included, must be an integer. If the \(\text{order}\) is less than zero, the result will be an anti-derivative.

\(d()\) does not follow the normal evaluation mechanism of fully simplifying its arguments and then applying the function definition to these fully simplified arguments. Instead, \(d()\) performs the following steps:

1. Simplify the second argument only to the extent that it does not lead to a non-variable.
2. Simplify the first argument only to the extent that it does recall any stored value for the variable determined by step 1.
3. Determine the symbolic derivative of the result of step 2 with respect to the variable from step 1.
4. If the variable from step 1 has a stored value or a value specified by a "with" (|) operator, substitute that value into the result from step 3.

\[d\left(f(x) \cdot g(x), x\right) \Rightarrow f(x) \cdot g(x) + g(x) \cdot f'(x)\]
\[d\left(x^2 \cdot y^3, x, y\right) \Rightarrow 6 \cdot y^2 \cdot x\]
\[d\left(x^2 \cdot x^{-1}\right) \Rightarrow x^3 \div 3\]
\[d\left(x^2, x^3, x^4\right) \Rightarrow \{2 \cdot x^3, 3 \cdot x^2, 4 \cdot x^3\}\]

\(\text{data} \text{mat}\)  CATALOG/MATH/List menu

\[\text{data} \text{mat} \{\text{data1[,row1[,col1[,row2[,col2]]]]}\} \Rightarrow \text{data} \text{mat} \{\text{data1[,ml1[,1,1]]}\} \Rightarrow \text{data} \text{mat} \{\text{d1,ml1,1,1}\} \Rightarrow \text{Done}\]

Converts data to a matrix.

Each argument \{\text{row1[,col1[,row2[,col2]]]]\} can be individually omitted. If \text{row1} is omitted the default is 1. If \text{col1} is omitted the default is 1. If \text{row2} is omitted, the default is "max row." If \text{col2} is omitted, the default is "max column."

The \text{DATA} structure allows empty cells. Rows do not have to be equal size. When data is saved as a matrix, empty cells will be populated with "undef."

\(\text{dayOfWk()}\)  CATALOG

\[\text{dayOfWk}(\text{year}, \text{month}, \text{day}) \Rightarrow \text{integer}\]

\[\text{dayOfWk}(1948, 9, 6) \Rightarrow 2\]

Returns an integer from 1 to 7, with each integer representing a day of the week. Use \text{dayOfWk()} to determine on which day of the week a particular date would occur.

\(\text{Note}\): May not give accurate results for years

\[\begin{align*}
1 & = \text{Sunday} \\
2 & = \text{Monday}
\end{align*}\]
Enter the year as a four-digit integer. The month and day can be either one- or two-digit integers.

3 = Tuesday
4 = Wednesday
5 = Thursday
6 = Friday
7 = Saturday

**DD** MATH/Angle menu

<table>
<thead>
<tr>
<th>number/DD ⇒ value</th>
<th>list/DD ⇒ list</th>
<th>matrix/DD ⇒ matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the decimal equivalent of the argument expressed in degrees. The argument is a number, list, or matrix that is interpreted by the Mode setting in gradians, radians or degrees.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In Degree angle mode:

1.5° DD ENTER 1.5°

45° 22'14.3" DD ENTER 45.370...

{45° 22'14.3", 60° 0'0"} DD ENTER {45.370... 60}°

In Gradian angle mode:

9/10° DD ENTER (9/10)°

In Radian angle mode:

1.5 DD ENTER 85.9°

**Dec** MATH/Base menu

<table>
<thead>
<tr>
<th>integer/DD ⇒ integer</th>
<th>0b10011 DD ENTER 19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Converts integer to a decimal (base 10) number. A binary or hexadecimal entry must always have a 0b or 0h prefix, respectively.</td>
<td></td>
</tr>
</tbody>
</table>

- Zero, not the letter O, followed by b or h.

<table>
<thead>
<tr>
<th>ob binaryNumber</th>
<th>0b</th>
</tr>
</thead>
<tbody>
<tr>
<td>0h hexadecimalNumber</td>
<td></td>
</tr>
</tbody>
</table>

- A binary number can have up to 32 digits. A hexadecimal number can have up to 8.

Without a prefix, integer is treated as decimal. The result is displayed in decimal, regardless of the Base mode.
### Define CATALOG

**Define** `funcName(arg1Name, arg2Name, ...) = expression`  

Creates `funcName` as a user-defined function. You then can use `funcName()`, just as you use built-in functions. The function evaluates `expression` using the supplied arguments and returns the result.

- `funcName` cannot be the name of a system variable or built-in function.
- The argument names are placeholders; you should not use those same names as arguments when you use the function.

**Note:** This form of `Define` is equivalent to executing the expression: `expression!funcName(arg1Name,arg2Name)`.  
This command also can be used to define simple variables; for example, `Define a=3`.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define <code>g(x,y)=2x-3y</code></td>
<td><code>Done</code></td>
</tr>
<tr>
<td><code>g(1.2)</code></td>
<td><code>-4</code></td>
</tr>
<tr>
<td><code>1+a:2+b:g(a,b)</code></td>
<td><code>-4</code></td>
</tr>
<tr>
<td>Define <code>h(x)=when(x&lt;2,2x&lt;3,-2x+3)</code></td>
<td><code>Done</code></td>
</tr>
<tr>
<td><code>h(-3)</code></td>
<td><code>-9</code></td>
</tr>
<tr>
<td><code>h(4)</code></td>
<td><code>-5</code></td>
</tr>
<tr>
<td>Define <code>eigenvl(a)=cZeros(det(identity(dim(a))[[1]]-x*â•ã‚»a),x)</code></td>
<td><code>Done</code></td>
</tr>
<tr>
<td><code>eigenvl([-1.2;4;3])</code></td>
<td><code>{2 \sqrt{3} - 1 \over 11} - (2\sqrt{3} + 1)_{11}</code></td>
</tr>
</tbody>
</table>

**Define** `funcName(arg1Name, arg2Name, ...) = Func block EndFunc`  

Is identical to the previous form of `Define`, except that in this form, the user-defined function `funcName()` can execute a block of multiple statements.

- `block` can be either a single statement or a series of statements separated with the `:` character.  
- `block` also can include expressions and instructions (such as `If`, `Then`, `Else`, and `For`).  
This allows the function `funcName()` to use the `Return` instruction to return a specific result.

**Note:** It is usually easier to author and edit this form of Function in the program editor rather than on the entry line.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define <code>g(x,y)=Func:If x&gt;y Then:Return x:Else:Return y:EndIf:EndFunc</code></td>
<td><code>Done</code></td>
</tr>
<tr>
<td><code>g(3,7)</code></td>
<td><code>3</code></td>
</tr>
</tbody>
</table>

**Define** `progName(arg1Name, arg2Name, ...) = Prgm block EndPrgm`  

Creates `progName` as a program or subprogram, but cannot return a result using `Return`. Can execute a block of multiple statements.

- `block` can be either a single statement or a series of statements separated with the `:` character.  
- `block` also can include expressions and instructions (such as `If`, `Then`, `Else`, and `For`) without restrictions.

**Note:** It is usually easier to author and edit a program block in the Program Editor rather than on the entry line.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define <code>listinpt()=prgm:Local n,i,str1,num:InputStr &quot;Enter name of list&quot;,str1:Input &quot;No. of elements&quot;,n:For i,1,n,1:Input &quot;element &amp;string(i),num: num#str1[i]:EndFor:EndPrgm</code></td>
<td><code>Done</code></td>
</tr>
<tr>
<td><code>listinpt()</code></td>
<td><code>Enter name of list</code></td>
</tr>
</tbody>
</table>

### DelFold CATALOG

**DelFold folderName1[, folderName2[, folderName3[ ...]]**  

Deletes user-defined folders with the names `folderName1, folderName2` etc. An error message is displayed if the folders contain any variables.

**Note:** You cannot delete the main folder.
**DelType**

**DelType var_type**

Deletes all unlocked variables of the type specified by var_type.

**Note:** Possible values for var_type are:

ASM, DATA, EXPR, FUNC, GDB, LIST, MAT, PIC, PRGM, STR, TEXT, AppVar_type_name, All.

---

**DelVar**

**CATALOG**

**DelVar var1[, var2[, var3]]...**

Deletes the specified variables from memory.

---

**deSolve()** MATH/Calculus menu

**deSolve(1stOrderOde and initialCondition, independentVar, dependentVar)** ⇒ a general solution

Returns an equation that explicitly or implicitly specifies a general solution to the 1st- or 2nd-order ordinary differential equation (ODE). In the ODE:

- Use a prime symbol (’), press \( 2 \times \) to denote the 1st derivative of the dependent variable with respect to the independent variable.
- Use two prime symbols to denote the corresponding second derivative.

The ’ symbol is used for derivatives within deSolve() only. In other cases, use \( d() \).

The general solution of a 1st-order equation contains an arbitrary constant of the form \( @k \), where \( k \) is an integer suffix from 1 through 255. The suffix resets to 1 when you use ClrHome or Press \( 8 \): Clear Home. The solution of a 2nd-order equation contains two such constants.

Apply solve() to an implicit solution if you want to try to convert it to one or more equivalent explicit solutions.

When comparing your results with textbook or manual solutions, be aware that different methods introduce arbitrary constants at different points in the calculation, which may produce different general solutions.

---

When comparing your results with textbook or manual solutions, be aware that different methods introduce arbitrary constants at different points in the calculation, which may produce different general solutions.

---

**deSolve(1stOrderOde and initialCondition, independentVar, dependentVar)** ⇒ a particular solution

Returns a particular solution that satisfies 1stOrderOde and initialCondition. This is usually easier than determining a general solution, substituting initial values, solving for the arbitrary constant, and then substituting that value into...
the general solution.

initialCondition is an equation of the form:

dependentVar (initialIndependentValue) = initialDependentValue

The initialIndependentValue and initialDependentValue can be variables such as x0 and y0 that have no stored values. Implicit differentiation can help verify implicit solutions.

soln|x=0 and y=0 :
true
d(right(eq)-left(eq),x)/((left(eq)+right(eq),y)) impdif(eq,x,y) ENTER

Done

do|y'=impdif(soln,x,y) ENTER

true

DelVar do,soln

Done

desolve(
2ndOrderOde and initialCondition1 and 
initialCondition2, independentVar,
dependentVar)
⇒⇒⇒⇒
a particular solution

Returns a particular solution that satisfies 2ndOrderOde and has specified values of the dependent variable and its first derivative at one point.

For initialCondition1, use the form:

dependentVar (initialIndependentValue) = initialDependentValue

For initialCondition2, use the form:

dependentVar' (initialIndependentValue) = initial1stDerivativeValue

desolve(
2ndOrderOde and boundaryCondition1 and 
boundaryCondition2, independentVar,
dependentVar)
⇒⇒⇒⇒
a particular solution

Returns a particular solution that satisfies 2ndOrderOde and has specified values at two different points.

MATH/Matrix menu

det(squareMatrix, tol) ⇒ expression

Returns the determinant of squareMatrix.

Optionally, any matrix element is treated as zero if its absolute value is less than tol. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, tol is ignored.

• If you use ENTER or set the mode to Exact/Approx=APPROXIMATE, computations are done using floating-point arithmetic.

• If tol is omitted or not used, the default tolerance is calculated as:

\[ \text{tol} = \max(\text{dim(squareMatrix)}) \times \text{rowNorm(squareMatrix)} \]

Appendix A: Functions and Instructions
### diag()

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>diag(rowMatrix)</td>
<td>matrix</td>
</tr>
<tr>
<td>diag(columnMatrix)</td>
<td>matrix</td>
</tr>
</tbody>
</table>

Returns a matrix with the values in the argument list or matrix in its main diagonal.

### Dim()

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>dim(list)</td>
<td>integer</td>
</tr>
<tr>
<td>dim(matrix)</td>
<td>list</td>
</tr>
<tr>
<td>dim(string)</td>
<td>integer</td>
</tr>
</tbody>
</table>

Returns the dimension of list, the dimensions of matrix as a two-element list (rows, columns), and the number of characters contained in character string string.
Appendix A: Functions and Instructions

Disp CATALOG

Disp [exprOrString1], [exprOrString2] ... Displays the current contents of the Program I/O screen. If one or more exprOrString is specified, each expression or character string is displayed on a separate line of the Program I/O screen.

An expression can include conversion operations such as DDD and Rect. You can also use the \ operator to perform unit and number base conversions.

If Pretty Print = ON, expressions are displayed in pretty print.

From the Program I/O screen, you can press ‡ to display the Home screen, or a program can use DispHome.

Note: To type an underscore ( _ ), press ¥

To type \, press [ [ ] ]

DispG CATALOG

DispG Displays the current contents of the Graph screen.

In function graphing mode:

Program segment:

\[ y_1(x) = 5 \cos(x) \]

\[ \text{DispG} \]

DispHome CATALOG

DispHome Displays the current contents of the Home screen.

Program segment:

\[ \text{DispG} \]

DispTbl CATALOG

DispTbl Displays the current contents of the Table screen.

Note: The cursor pad is active for scrolling. Press (ESC) or (ENTER) to resume execution if in a program.

Note: To type an underscore ( _ ), press ¥

To type \, press [ [ ] ]

Disp "Hello" [ENTER] Hello

Disp cos(2.3) [ENTER] -.666...

{1,2,3,4} [ENTER] L1 [ENTER] (1 2 3 4)

Disp 180_min [ENTER] _hr [ENTER] 3. _hr

Disp "Hello"

Disp cos(2.3)

Disp {1,2,3,4}

Disp 180_min _hr

Note: To type an underscore ( _ ), press [ [ ] ]

To type \, press [ [ ] ]

DispG

Disp "Hello"

Disp cos(2.3)

Disp {1,2,3,4}

Disp 180_min _hr
DMS  MATH/Angle menu

expression DMS
list DMS
matrix DMS

Interprets the argument as an angle and displays the equivalent DMS (DDD.DDDD  ¡  MM  ¢  SS.ss £) number.
See "", ¢ on page 910 for DMS (degree, minutes, seconds) format.

Note: DMS will convert from radians to degrees when used in radian mode. If the input is followed by a degree symbol ( ¡ ), no conversion will occur. You can use DMS only at the end of an entry line.

<table>
<thead>
<tr>
<th>MATH/Matrix/Vector ops menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>dotP(list1, list2)  expression</td>
</tr>
<tr>
<td>Returns the &quot;dot&quot; product of two lists.</td>
</tr>
<tr>
<td>dotP(vector1, vector2)  expression</td>
</tr>
<tr>
<td>Returns the &quot;dot&quot; product of two vectors.</td>
</tr>
</tbody>
</table>

DrawFunc  CATALOG

DrawFunc expression

Draws expression as a function, using x as the independent variable.

Note: Regraphing erases all drawn items.

DrawInv  CATALOG

DrawInv expression

Draws the inverse of expression by plotting x values on the y axis and y values on the x axis.

x is the independent variable.

Note: Regraphing erases all drawn items.
DrawParm  

**CATALOG**

**DrawParm**  

**expression1, expression2** 

\[ t, t_{\text{min}}, t_{\text{max}}, t_{\text{step}} \]

Draws the parametric equations \( \text{expression1} \) and \( \text{expression2} \), using \( t \) as the independent variable.

Defaults for \( t_{\text{min}}, t_{\text{max}}, \) and \( t_{\text{step}} \) are the current settings for the Window variables \( t_{\text{min}}, t_{\text{max}}, \) and \( t_{\text{step}} \). Specifying values does not alter the window settings. If the current graphing mode is not parametric, these three arguments are required.

**Note:** Regraphing erases all drawn items.

In function graphing mode and ZoomStd window:

\[ \text{DrawParm } t \cdot \cos(t), t \cdot \sin(t), 0.10, 1 \]

Draws the parametric equations \( t \cdot \cos(t) \) and \( t \cdot \sin(t) \) from 0 to 1 using \( t \) as the independent variable.

\[ \text{Enter} \]

**DrawPol**  

**CATALOG**

**DrawPol**  

**expression\( q \), \( q_{\text{min}}, \( q_{\text{max}}, \( q_{\text{step}} \)

Draws the polar graph of \( \text{expression} \), using \( q \) as the independent variable.

Defaults for \( q_{\text{min}}, q_{\text{max}}, \) and \( q_{\text{step}} \) are the current settings for the Window variables \( q_{\text{min}}, q_{\text{max}}, \) and \( q_{\text{step}} \). Specifying values does not alter the window settings. If the current graphing mode is not polar, these three arguments are required.

**Note:** Regraphing erases all drawn items.

In function graphing mode and ZoomStd window:

\[ \text{DrawPol } 5 \cdot \cos(3 \cdot q), 0, 3.5, 0.1 \]

Draws the polar graph of \( 5 \cdot \cos(3 \cdot q) \) from 0 to 3.5 using \( q \) as the independent variable.

\[ \text{Enter} \]

**DrawSlp**  

**CATALOG**

**DrawSlp**  

**x1, y1, slope**

Displays the graph and draws a line using the formula \( y = y_1 + \text{slope} \cdot (x - x_1) \).

**Note:** Regraphing erases all drawn items.

In function graphing mode and ZoomStd window:

\[ \text{DrawSlp } 2, 3, -2 \]

\[ \text{Enter} \]

**DropDown**  

**CATALOG**

**DropDown**  

**titleString, \{item1String, item2String, ...\}, \text{varName}**

Displays a drop-down menu with the name \( \text{titleString} \) and containing the items \( 1: \text{item1String} \), \( 2: \text{item2String} \), and so forth. **DropDown** must be within a **Dialog...EndDialog** block.

If \( \text{varName} \) already exists and has a value within the range of items, the referenced item is displayed as the default selection. Otherwise, the menu’s first item is the default selection.

When you select an item from the menu, the corresponding number of the item is stored in the variable \( \text{varName} \). (If necessary, **DropDown** creates \( \text{varName} \).)

See **Dialog** program listing example.
DrwCtour draws contours on the current 3D graph at the z values specified by expression or list. The 3D graphing mode must already be set. DrwCtour automatically sets the graph format style to CONTOUR LEVELS.

By default, the graph automatically contains the number of equally spaced contours specified by the ncontour Window variable. DrwCtour draws contours in addition to the defaults. To turn off the default contours, set ncontour to zero, either by using the Window screen or by storing 0 to the ncontour system variable.

In 3D graphing mode:

\[ (1/5)x^2+(1/5)y^2-10z(x,y) \]

Use the cursor to change the viewing angle. Press 0 (zero) to return to the original view.

To toggle between different graph format styles, press \[ \text{¥} \].

Press \( X, Y, \) or \( Z \) to look down the corresponding axis.

### E key

#### \( \text{mantissa} \times 10^{\text{exponent}} \)

Enters a number in scientific notation. The number is interpreted as \( \text{mantissa} \times 10^{\text{exponent}} \).

**Hint:** If you want to enter a power of 10 without causing a decimal value result, use \( 10^{\text{integer}} \).

#### \( e^x() \) key

Returns \( e \) raised to the \( x \) power.

**Note:** On the TI-89 Titanium, pressing \( \text{[E]} \) to display \( e^x \) is different from pressing \( \text{[E]} \).

On the Voyage 200, pressing \( \text{[E]} \) to display \( e^x \) is different from accessing the character \( e \) from the QWERTY keyboard.

You can enter a complex number in \( r \text{e}^{\text{i}\theta} \) form. However, use this form in Radian angle mode only; it causes a Domain Error in Degree or Gradian angle mode.

#### \( e^x() \) \( \Rightarrow \) \( \text{list} \)

Returns \( e \) raised to the power of each element in \( \text{list} \).

### \( e^x() \) \( \Rightarrow \) \( \text{squareMatrix} \)

Returns the matrix exponential of \( \text{squareMatrix} \).

This is not the same as calculating \( e \) raised to the power of each element. For information about the calculation method, refer to \( \text{cos}() \).

\( \text{squareMatrix} \) must be diagonalizable. The result always contains floating-point numbers.

\[ e^{(1,5.3;4.2,1.6,\cdot,2.1)} \]
eigVc()  MATH/Matrix menu

eigVc(squareMatrix) ⇒ matrix

Returns a matrix containing the eigenvectors for a real or complex squareMatrix, where each column in the result corresponds to an eigenvalue. Note that an eigenvector is not unique; it may be scaled by any constant factor. The eigenvectors are normalized, meaning that if \( \mathbf{V} = [x_1, x_2, \ldots, x_n] \), then:

\[
\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2} = 1
\]

squareMatrix is first balanced with similarity transformations until the row and column norms are as close to the same value as possible. The squareMatrix is then reduced to upper Hessenberg form and the eigenvectors are computed via a Schur factorization.

In Rectangular complex format mode:

\[
\begin{bmatrix}
1.2 & 5.3 & -6.9 & 2.4 & -5.7
\end{bmatrix} \rightarrow m1 \quad \text{ENTER}
\begin{bmatrix}
1 & 2 & 5
3 & 6 & 9
2 & 5 & 7
\end{bmatrix}
\]

eigVc(m1) \quad \text{ENTER}

\[
\begin{bmatrix}
0.800 & 0.767 & 0.767 & 0.484 & 0.573 & 0.052 & 0.573 & 0.573 & 0.052
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.352 & 0.262 & 0.096 & 0.262 & 0.096 & 0.052 & 0.052 & 0.052 & 0.052
\end{bmatrix}
\]

eigVl()  MATH/Matrix menu

eigVl(squareMatrix) ⇒ list

Returns a list of the eigenvalues of a real or complex squareMatrix.

squareMatrix is first balanced with similarity transformations until the row and column norms are as close to the same value as possible. The squareMatrix is then reduced to upper Hessenberg form and the eigenvalues are computed from the upper Hessenberg matrix.

In Rectangular complex format mode:

\[
\begin{bmatrix}
1.2 & 5.3 & -6.9 & 2.4 & -5.7
\end{bmatrix} \rightarrow m1 \quad \text{ENTER}
\begin{bmatrix}
1 & 2 & 5
3 & 6 & 9
2 & 5 & 7
\end{bmatrix}
\]

eigVl(m1) \quad \text{ENTER}

\[
\{4.409, 2.204 + 0.763 \text{i}, 2.204 - 0.763 \text{i}\}
\]

Else  See If, page 830.

ElseIf  CATALOG  See also If, page 830.

If  Boolean expression1 Then

block1
EndIf

ElseIf  Boolean expression2 Then

block2
EndIf

ElseIf  Boolean expressionN Then

blockN
EndIf

ElseIf  can be used as a program instruction for program branching.

EndCustm  See Custom, page 802.

EndDlog  See Dialog, page 810.

EndFor  See For, page 822.

EndFunc  See Func, page 823.

EndIf  See If, page 830.

Appendix A: Functions and Instructions  815
**entry()**

- **entry()** ⇒ expression
- **entry(integer)** ⇒ expression

Returns a previous entry-line entry from the Home screen history area.

- **integer**, if included, specifies which entry expression in the history area. The default is 1, the most recently evaluated entry. Valid range is from 1 to 99 and cannot be an expression.

**Note:** If the last entry is still highlighted on the Home screen, pressing [ENTER] is equivalent to executing `entry(1)`.

**On the Home screen:**

```
1+1/x [ENTER]  \( \frac{1}{x} + 1 \)
1+1/entry(1) [ENTER]  \( 2 - \frac{1}{x+1} \)
1/2*(2*entry(1)) [ENTER]  \( \frac{1}{2} \cdot (2 \cdot \frac{1}{x+1}) + \frac{3}{2} \)
5/3-1/3*entry(2) [ENTER]  \( \frac{5}{3} - \frac{1}{3 \cdot (3 \cdot \frac{1}{x+2})} \)
entry(4) [ENTER]  \( \frac{1}{x} + 1 \)
```

**exact()**

- **exact(expression1[, tol])** ⇒ expression
- **exact(list1[, tol])** ⇒ list
- **exact(matrix1[, tol])** ⇒ matrix

Uses Exact mode arithmetic regardless of the Exact/Approx mode setting to return, when possible, the rational-number equivalent of the argument.

- **tol** specifies the tolerance for the conversion; the default is 0 (zero).

**Examples:**

```
effect(.25) [ENTER]  \( 1/4 \)
effect(.333333) [ENTER]  \( \frac{333333}{1000000} \)
effect(.33333,.001) [ENTER]  \( 1/3 \)
effect(3.5x+y) [ENTER]  \( \frac{7 \cdot x + y}{2} \)
effect(.2,.334.125) [ENTER]  \{ 1/5  \frac{33}{100}  \frac{33}{8} \}
```
**Exec**  

**CATALOG**

`Exec string1, expression1 [, expression2] ...`

Executes a `string1` consisting of a series of Motorola 68000 op-codes. These codes act as a form of an assembly-language program. If needed, the optional `expression1` let you pass one or more arguments to the program.

For more information, check the TI Web site:  

**Warning:** `Exec` gives you access to the full power of the microprocessor. Please be aware that you can easily make a mistake that locks up the calculator and causes you to lose your data. We suggest you make a backup of the calculator contents before attempting to use the `Exec` command.

**Exit**  

**CATALOG**

`Exit`

Exits the current `For`, `While`, or `Loop` block.  

`Exit` is not allowed outside the three looping structures (`For`, `While`, or `Loop`).

**expList()**  

**CATALOG**

`expList(expression, var) ⇒ list`

Examines `expression` for equations that are separated by the word “or,” and returns a list containing the right-hand sides of the equations of the form `var=expression`. This gives you an easy way to extract some solution values embedded in the results of the `solve()`, `cSolve()`, `fMin()`, and `fMax()` functions.

**Note:** `expList()` is not necessary with the `zeros` and `cZeros()` functions because they return a list of solution values directly.

**expand()**  

**MATH/Algebra menu**

`expand(expression1 [, var]) ⇒ expression`

`expand(list1 [, var]) ⇒ list`

`expand(matrix1 [, var]) ⇒ matrix`

`expand(expression1)` returns `expression1` expanded with respect to all its variables. The expansion is polynomial expansion for polynomials and partial fraction expansion for rational expressions.

The goal of `expand()` is to transform `expression1` into a sum and/or difference of simple terms. In contrast, the goal of `factor()` is to transform `expression1` into a product and/or quotient of simple factors.

---

Appendix A: Functions and Instructions  
817
expand(expression1, var) returns expression expanded with respect to var. Similar powers of var are collected. The terms and their factors are sorted with var as the main variable. There might be some incidental factoring or expansion of the collected coefficients. Compared to omitting var, this often saves time, memory, and screen space, while making the expression more comprehensible.

\[
\text{expand((x+y+1)^2, y)}
\]

\[
\begin{align*}
(x+y+1)^2 &= x^2 + 2xy + y^2 + 2x + 2y + 1 \\
&= (x+1)^2 + (y+1)^2
\end{align*}
\]

\[
\text{expand((x+y+1)^2, x)}
\]

\[
\begin{align*}
(x+y+1)^2 &= x^2 + 2xy + y^2 + 2x + 2y + 1 \\
&= (x+1)^2 + (y+1)^2
\end{align*}
\]

Even when there is only one variable, using var might make the denominator factorization used for partial fraction expansion more complete.

**Hint:** For rational expressions, propFrac() is a faster but less extreme alternative to expand().

**Note:** See also comDenom() for an expanded numerator over an expanded denominator.

\[
\text{expr(string)} \Rightarrow \text{expression}
\]

Returns the character string contained in string as an expression and immediately executes it.

\[
\text{expr("1+2*x+2*x")}
\]

\[
x^2 + x + 3
\]

\[
\text{expr("expand((1+x)/2")}
\]

\[
x^2 + 2x + 1
\]

"Define cube(x)=x^3"→funcstr

"Define cube(x)=x^3"

expr(funcstr) → Done

cube(2) → 8
ExpReg  MATH/Statistics/Regressions menu

ExpReg(list1, list2[, list3[, list4, list5]]
Calculates the exponential regression and updates all the system statistics variables.
For all list, list2 represents ylist.
list3 represents frequency.
list4 represents category codes.
list5 represents category include list.
Note: list1 through list4 must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). list5 does not have to be a variable name and cannot be c1–c99.

In function graphing mode:

{1,2,3,4,5,6,7,8}→L1 ENTER (1 2 ...)
{1,2,2,3,4,5,7}→L2 ENTER (1 2 ...)
ExpReg L1,L2 ENTER Done
ShowStat ENTER

Regeq(x)
"y1(x)
Done
NewPlot 1,1,L1,L2 ENTER Done

factor()  MATH/Algebra menu

factor(expression1, var) ⇒ expression
factor(list1, var) ⇒ list
factor(matrix1, var) ⇒ matrix

factor(expression1) returns expression1 factored with respect to all of its variables over a common denominator.
expression1 is factored as much as possible toward linear rational factors without introducing new non-real subexpressions. This alternative is appropriate if you want factorization with respect to more than one variable.

factor(expression1, var) returns expression1 factored with respect to variable var.
expression1 is factored as much as possible toward real factors that are linear in var, even if it introduces irrational constants or subexpressions that are irrational in other variables.
The factors and their terms are sorted with var as the main variable. Similar powers of var are collected in each factor. Include var if factorization is needed with respect to only that variable and you are willing to accept irrational expressions in any other variables to increase factorization with respect to var. There might be some incidental factoring with respect to other variables.

factor(expression1, var) = expression
factor(list1, var) = list
factor(matrix1, var) = matrix
For the AUTO setting of the Exact/Approx mode, including \texttt{var} permits approximation with floating-point coefficients where irrational coefficients cannot be explicitly expressed concisely in terms of the built-in functions. Even when there is only one variable, including \texttt{var} might yield more complete factorization.

**Note:** See also \texttt{comDenom()} for a fast way to achieve partial factoring when \texttt{factor()} is not fast enough or if it exhausts memory.

**Note:** See also \texttt{cFactor()} for factoring all the way to complex coefficients in pursuit of linear factors.

\begin{verbatim}
factor(x^5+4x^4+5x^3-6x-3) \quad \text{ENTER}
\end{verbatim}
\begin{equation}
x^5 + 4x^4 + 5x^3 - 6x - 3
\end{equation}

\begin{verbatim}
factor(ans(1),x) \quad \text{ENTER}
\end{verbatim}
\begin{equation}
(x-.964...) \cdot (x+6.11...) \cdot \\
(x+2.125...) \cdot (x^2 + 2.227...) \\
\quad (x+2.392...)
\end{equation}

\textbf{factor(\texttt{rationalNumber})} returns the rational number factored into primes. For composite numbers, the computing time grows exponentially with the number of digits in the second-largest factor. For example, factoring a 30-digit integer could take more than a day, and factoring a 100-digit number could take more than a century.

**Note:** To stop (break) a computation, press \texttt{ON}.

If you merely want to determine if a number is prime, use \texttt{isPrime()} instead. It is much faster, particularly if \texttt{rationalNumber} is not prime and if the second-largest factor has more than five digits.

\begin{verbatim}
factor(152417172689) \quad \text{ENTER}
\end{verbatim}
\begin{equation}
123457 \cdot 1234577
\end{equation}

\begin{verbatim}
isPrime(152417172689) \quad \text{ENTER}
\end{verbatim}
\begin{equation}
\text{false}
\end{equation}

\begin{verbatim}
floor() \quad \text{MATH/Number menu}
\end{verbatim}

\begin{verbatim}
floor(\texttt{expression}) \Rightarrow \text{integer}
floor(-2.14) \quad \text{ENTER}
\end{verbatim}
\begin{equation}
\text{-3.}
\end{equation}

Returns the greatest integer that is \leq the argument. This function is identical to \texttt{int()}.

The argument can be a real or a complex number.

\begin{verbatim}
floor(\texttt{list}) \Rightarrow \text{list}
floor(\{3/2.0,-5.3\}) \quad \text{ENTER}
\end{verbatim}
\begin{equation}
\{1 \text{. } -6.\}
\end{equation}

Returns a list or matrix of the floor of each element.

**Note:** See also \texttt{ceiling()} and \texttt{int()}.
Appendix A: Functions and Instructions

fMax()  MATH/Calculus menu

\[ fMax(expression, var) \Rightarrow \text{Boolean expression} \]

Returns a Boolean expression specifying candidate values of \( var \) that maximize \( expression \) or locate its least upper bound.

\[ fMax(1- (x-a)^2- (x-b)^2,x) \]

\[ x = \frac{a+b}{2} \]

\[ fMax(.5x^3- x-2,x) \]

\[ x = \infty \]

Use the “|” operator to restrict the solution interval and/or specify the sign of other undefined variables.

\[ fMax(.5x^3- x-2,x)|x \leq 1 \]

\[ x = .816... \]

For the APPROX setting of the Exact/Approx mode, \( fMax() \) iteratively searches for one approximate local maximum. This is often faster, particularly if you use the “|” operator to constrain the search to a relatively small interval that contains exactly one local maximum.

Note: See also \( fMin() \) and \( max() \).

fMin()  MATH/Calculus menu

\[ fMin(expression, var) \Rightarrow \text{Boolean expression} \]

Returns a Boolean expression specifying candidate values of \( var \) that minimize \( expression \) or locate its greatest lower bound.

\[ fMin(1- (x-a)^2- (x-b)^2,x) \]

\[ x = \infty \text{ or } x = -\infty \]

\[ fMin(.5x^3- x-2,x)|x = 1 \]

\[ x = 1 \]

\[ fMin(a* x^2,x) \]

\[ x = 0 \text{ or } a = 0 \]

\[ fMin(a* x^2,x)|a>0 \text{ and } x>1 \]

\[ x = 1 \]

\[ fMin(a* x^2,x)|a>0 \]

\[ x = 0 \]

Use the “|” operator to restrict the solution interval and/or specify the sign of other undefined variables.

For the APPROX setting of the Exact/Approx mode, \( fMin() \) iteratively searches for one approximate local minimum. This is often faster, particularly if you use the “|” operator to constrain the search to a relatively small interval that contains exactly one local minimum.

Note: See also \( fMax() \) and \( min() \).

FnOff  CATALOG

Deselects all \( Y= \) functions for the current graphing mode.

In split-screen, two-graph mode, \( FnOff \) only applies to the active graph.

\[ FnOff \]

Deselects the specified \( Y= \) functions for the current graphing mode.

In function graphing mode:

\[ FnOff 1,3 \]

deselects \( y1(x) \) and \( y3(x) \).

In parametric graphing mode:

\[ FnOff 1,3 \]

deselects \( xt1(t), yt1(t), xt3(t), \) and \( yt3(t) \).
**FnOn**  
**CATALOG**

**FnOn**

Selects all Y= functions that are defined for the current graphing mode.

In split-screen, two-graph mode, **FnOn** only applies to the active graph.

**FnOn [1] [, 2] ... [,99]**

Selects the specified Y= functions for the current graphing mode.

**Note:** In 3D graphing mode, only one function at a time can be selected. **FnOn 2** selects z2(x,y) and deselects any previously selected function. In the other graph modes, previously selected functions are not affected.

**For**  
**CATALOG**

**For var, low, high [, step] block**

**EndFor**

Executes the statements in **block** iteratively for each value of **var**, from **low** to **high**, in increments of **step**.

**var** must not be a system variable.

**step** can be positive or negative. The default value is 1.

**block** can be either a single statement or a series of statements separated with the ":" character.

---

**format()**  
**MATH/String menu**

**format(expression, formatString) ⇒ string**

Returns **expression** as a character string based on the format template.

**expression** must simplify to a number. **formatString** is a string and must be in the form: "F[n]", "S[n]", "E[n]", "G[n][c]", where [ ] indicate optional portions.

F[n]: Fixed format. **n** is the number of digits to display after the decimal point.

S[n]: Scientific format. **n** is the number of digits to display after the decimal point.

E[n]: Engineering format. **n** is the number of digits after the first significant digit. The exponent is adjusted to a multiple of three, and the decimal point is moved to the right by zero, one, or two digits.

**Program segment:**

```plaintext
:tempsum :1 :step
:For i,1,100,step
: tempsum+i
:EndFor
:Disp tempsum
```

**Contents of tempsum after execution:**

5050

**Contents of tempsum when step is changed to 2:**

2500

**Examples:**

- `format(1.234567,"f3")` ➤ `"1.235"`
- `format(1.234567,"s2")` ➤ `"1.23 e 0"`
- `format(1.234567,"e3")` ➤ `"1.235 e 0"`
- `format(1.234567,"g3")` ➤ `"1.235"`
- `format(1234.567,"g3")` ➤ `"1.234.567"`
- `format(1.234567,"g3.0")` ➤ `"1.235"`
$G\text{n}c$: Same as fixed format but also separates digits to the left of the radix into groups of three. $c$ specifies the group separator character and defaults to a comma. If $c$ is a period, the radix will be shown as a comma.

$[Rc]$: Any of the above specifiers may be suffixed with the $Rc$ radix flag, where $c$ is a single character that specifies what to substitute for the radix point.

### fPart() MATH/Number menu

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| `fPart(expression)` | Returns the fractional part of the argument. For a list or matrix, returns the fractional parts of the elements. The argument can be a real or a complex number. | `fPart(1.234)`
|                 |                                                       | `{0.3, 0.003}`   |

In function graphing mode, define a piecewise function:

```
Define g(x)=Func:If x<0 Then:Return 3*cos(x):Else:Return 3*x:EndIf:EndFunc
```

Graph `g(x)`

### gcd() MATH/Number menu

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| `gcd(number1, number2)` | Returns the greatest common divisor of the two arguments. The gcd of two fractions is the gcd of their numerators divided by the lcm of their denominators. In Auto or Approximate mode, the gcd of fractional floating-point numbers is 1.0. | `gcd(18,33)`
|                 |                                                       | `{3, 7, 1}`      |

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>gcd(list1, list2)</code></td>
<td>Returns the greatest common divisors of the corresponding elements in <code>list1</code> and <code>list2</code>.</td>
<td><code>gcd({12,14,16}, {9,7,5})</code></td>
</tr>
<tr>
<td><code>gcd(matrix1, matrix2)</code></td>
<td>Returns the greatest common divisors of the corresponding elements in <code>matrix1</code> and <code>matrix2</code>.</td>
<td><code>gcd([2,4;6,8],[4,8;12,16])</code></td>
</tr>
</tbody>
</table>
Get

Get var

Retrieves a CBL 2™ (Calculator-Based Laboratory™) or CBR™ (Calculator-Based Ranger™) value from the link port and stores it in variable var.

Program segment:

```
:Send {3,1,-1,0}
:For i,1,99
  :Get data[i]
  :PtOn i,data[i]
:EndFor
```

GetCalc

GetCalc var

Retrieves a value from the link port and stores it in variable var. This is for unit-to-unit linking.

Note: To get a variable to the link port from another unit, use 2° on the other unit to select and send a variable, or do a SendCalc on the other unit.

Program segment:

```
:Disp "Press Enter when ready"
:Pause
:GetCalc L1
:Disp "List L1 received"
```

GetCalc var,port

Retrieves a value from the link port and stores it in variable var on the receiving TI-89 Titanium.

If the port is not specified, or port = 0 is specified, the TI-89 Titanium waits for data from either port.

If port = 1, the TI-89 Titanium waits for data from the USB port.

If port = 2, the TI-89 Titanium waits for data from the I/O port.

dgetConfig()

dgetConfig() ⇒ ListPairs

Returns a list of calculator attributes. The attribute name is listed first, followed by its value.

```
getConfig() ⇒ enter

("Product Name" "Advanced Mathematics Software"
 "Version" "2.00, 09/25/1999"
 "Product ID" "03-1-4-68"
 "ID #" "01012 34567 ABCD"
 "Cert. Rev. #" 0
 "Screen Width" 160
 "Screen Height" 100
 "Window Width" 160
 "Window Height" 67
 "RAM Size" 262132
 "Free RAM" 197178
 "Archive Size" 655360
 "Free Archive" 655340)
```

Note: Your screen may display different attribute values. The Cert. Rev. # attribute appears only if you have purchased and installed additional software into the calculator.

dateGet()

dateGet() ⇒ list

Returns a list giving the date according to the current value of the clock. The list is in {year,month,day} format.

```
dateGet() ⇒ enter

{2002 2 22}
```
**getDenom()**  MATH/Algebra/Extract menu

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>getDenom((x^2)/(y-3))</td>
<td>y = 3</td>
</tr>
<tr>
<td>getDenom(2/7)</td>
<td>7</td>
</tr>
<tr>
<td>getDenom(1/x+(y^2+y)/y^2)</td>
<td>x \cdot y</td>
</tr>
</tbody>
</table>

**getDtFmt()**  CATALOG

- Integer values:
  - 1 = MM/DD/YY
  - 2 = DD/MM/YY
  - 3 = MM.DO.YY
  - 4 = DD.MM.YY
  - 5 = YY.MM.DD
  - 6 = MM-DD-YY
  - 7 = DD-MM-YY
  - 8 = YY-MM-DD

**getDtStr()**  CATALOG

- Optional integer values:
  - 1 = MM/DD/YY
  - 2 = DD/MM/YY
  - 3 = MM.DO.YY
  - 4 = DD.MM.YY
  - 5 = YY.MM.DD
  - 6 = MM-DD-YY
  - 7 = DD-MM-YY
  - 8 = YY-MM-DD

**getFold()**  CATALOG

- getFold() | "main"
- getFold( )oldfoldr | "main"

**getKey()**  CATALOG

- Program listing:
  ```plaintext
  :Disp
  :Loop
  :   getKey( )key
  :   while key=0
  :      getKey( )key
  :    EndWhile
  :  Disp key
  :  If key = ord("a")
  :    Stop
  :EndLoop
  ```

Appendix A: Functions and Instructions 825
### getMode() CATALOG

**getMode(modeNameString) ⇒ string**

- If the argument is a specific mode name, returns a string containing the current setting for that mode.
- If the argument is "ALL", returns a list of string pairs containing the settings of all the modes. If you want to restore the mode settings later, you must store the `getMode("ALL")` result in a variable, and then use `setMode()` to restore the modes.

For a listing of mode names and possible settings, see `setMode()`.

**Note:** To set or return information about the Unit System mode, use `setUnits()` or `getUnits()` instead of `setMode()` or `getMode()`.

### getNum() MATH/Algebra/Extract menu

**getNum(expression1) ⇒ expression**

Transforms `expression1` into one having a reduced common denominator, and then returns its numerator.

### getTime() CATALOG

**getTime() ⇒ list**

Returns a list giving the time according to the current value of the clock. The list is in {hour, minute, second} format. The time is returned in the 24 hour format.

### getTmFmt() CATALOG

**getTmFmt() ⇒ integer**

Returns an integer representing the clock time format that is currently set on the device.

**Integer values:**
- 12 = 12 hour clock
- 24 = 24 hour clock

### getTmStr() CATALOG

**getTmStr(integer) ⇒ string**

Returns a string of the current clock time in the current time format.

If you enter the optional integer that corresponds to a clock time format, the string returns the current time in the specified format.

**Note:** Your screen may display different mode settings.
getTmZn() CATALOG

getTmZn() ⇒ integer

Returns an integer representing the time zone that is currently set on the device.

The returned integer represents the number of minutes the time zone is offset from Greenwich Mean Time (GMT), as established in Greenwich, England. For example, if the time zone is offset from GMT by two hours, the device would return 120 (minutes).

Integers for time zones west of GMT are negative.
Integers for time zones east of GMT are positive.

If Greenwich Mean Time is 14:07:07, it is:
8:07:07 a.m. in Denver, Colorado (Mountain Daylight Time)
(–360 minutes from GMT)
16:07:07 p.m. in Brussels, Belgium (Central European Standard Time)
(+120 minutes from GMT)

datatype time zones

getType() CATALOG

getType(var) ⇒ string

Returns a string indicating the data type of variable var.
If var has not been defined, returns the string "NONE".

Data Type | Variable Contents
---|---
"ASM" | Assembly-language program
"DATA" | Data type
"EXPR" | Expression (includes complex/arbitrary/undefined, \(\infty\), \(\infty\), TRUE, FALSE, \(\pi\), \(e\))
"FUNC" | Function
"GDB" | Graph data base
"LIST" | List
"MAT" | Matrix
"NONE" | Variable does not exist
"NUM" | Real number
"OTHER" | Miscellaneous data type for future use by software applications
"PIC" | Picture
"PRGM" | Program
"STR" | String
"TEXT" | Text type
"VAR" | Name of another variable
getUnits()  CATALOG

getUnits() ⇒ list

Returns a list of strings that contain the current default units for all categories except constants, temperature, amount of substance, luminous intensity, and acceleration. list has the form:

(“system” “cat1” “unit1” “cat2” “unit2” …)

The first string gives the system (SI, ENG/US, or CUSTOM). Subsequent pairs of strings give a category (such as Length) and its default unit (such as _m for meters).

To set the default units, use setUnits().

Note: Your screen may display different default units.

Goto  CATALOG

Goto  labelName

Transfers program control to the label labelName. labelName must be defined in the same program using a Lbl instruction.

Program segment:

© :0⇒temp
© :1⇒
© :Lbl TOP
: temp+i⇒temp
: If i<10 Then
: i+1⇒
: Goto TOP
: EndIf
: Disp temp
©

Grad  CATALOG/MATH/Angle menu

Grad expression

Converts an expression to gradian angle measure.

In Degree angle mode:

1.5 Grad ENTER  1.66667°

In Radian angle mode:

1.5 Grad ENTER  96.493°
Graph

Graph expression1[, expression2] [,, var1] [,, var2]

The Smart Graph feature graphs the requested expressions/ functions using the current graphing mode.

Expressions entered using the Graph or Table commands are assigned increasing function numbers starting with 1. They can be modified or individually deleted using the edit functions available when the table is displayed by pressing [Edit] Header. The currently selected Y= functions are ignored.

If you omit an optional var argument, Graph uses the independent variable of the current graphing mode.

Note: Not all optional arguments are valid in all modes because you can never have all four arguments at the same time.

Some valid variations of this instruction are:
- Function graphing
  \( \text{Graph} \ expr, x \)
- Parametric graphing
  \( \text{Graph} \ xExpr, yExpr, t \)
- Polar graphing
  \( \text{Graph} \ expr, \theta \)
- Sequence graphing
  Not allowed.
- 3D graphing
  \( \text{Graph} \ expr, x, y \)
- Diff Equations graphing
  Not allowed.

Note: Use ClrGraph to clear these functions, or go to the Y= Editor to re-enable the system Y= functions.

Hex

MATH/Base menu

\( \text{integer} \rightarrow \text{Hex} \rightarrow \text{integer} \)

Converts \( \text{integer} \) to a hexadecimal number. Binary or hexadecimal numbers always have a \( 0b \) or \( 0h \) prefix, respectively.

\( 0b \) or \( 0h \) followed by \( \text{binaryNumber} \) or \( \text{hexadecimalNumber} \)

A binary number can have up to 32 digits. A hexadecimal number can have up to 8.

Without a prefix, \( \text{integer} \) is treated as decimal (base 10). The result is displayed in hexadecimal, regardless of the Base mode.

If you enter a decimal integer that is too large for a signed, 32-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range.

Appendix A: Functions and Instructions 829
identity()  MATH/Matrix menu

identity(expression)  ➞  matrix

Returns the identity matrix with a dimension of expression.
expression must evaluate to a positive integer.

identity(4)  ➞

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

If

If Boolean expression Then  block  Else  block2  EndIf

If Boolean expression evaluates to true, executes block1 and then skips block2.
If Boolean expression evaluates to false, skips block1 but executes block2.
block1 and block2 can be a single statement.

imag()  MATH/Complex menu

imag(expression)  ➞  expression
imag(expression) returns the imaginary part of the argument.
Note: All undefined variables are treated as real variables. See also real().

imag(1+2i)  ➞  2
imag(z)  ➞  0
imag(x+y)  ➞  y

imag(list)  ➞  list
Returns a list of the imaginary parts of the elements.
imag([-3.4+1.1i])  ➞  {0 -1 1}
### Appendix A: Functions and Instructions

**imag(matrix)** ⇒ matrix

Returns a matrix of the imaginary parts of the elements.

\[
\text{imag([a,b;ic, id])} \quad \text{ENT} \\
\begin{bmatrix}
0 & 0 \\
\text{c} & \text{d}
\end{bmatrix}
\]

**ImpDif()** MATH/Calculus Menu, CATALOG

**ImpDif**(equation, independentVar, dependentVar[,order]) ⇒ expression

where the order defaults to 1.

Computes the implicit derivative for equations in which one variable is defined implicitly in terms of another.

\[
\text{impDif}(x^2+y^2=100,x,y) \quad \text{ENT} \\
-x/y
\]

**Indirection** See #(), page 908.

**Input** CATALOG

Pauses the program, displays the current Graph screen, and lets you update variables xc and yc (also r and q if for polar coordinate mode) by positioning the graph cursor.

When you press \( \text{ENT} \), the program resumes.

**Input** [promptString], var

Pauses the program, displays promptString on the Program I/O screen, waits for you to enter an expression, and stores the expression in variable var.

If you omit promptString, “?” is displayed as a prompt.

**InputStr** CATALOG

Pauses the program, displays promptString on the Program I/O screen, waits for you to enter a response, and stores your response as a string in variable var.

If you omit promptString, “?” is displayed as a prompt.

**Note:** The difference between Input and InputStr is that InputStr always stores the result as a string so that “ ” are not required.
inString()  MATH/String menu

inString(srcString, subString, start)  ⇒ integer

Returns the character position in string srcString at which the first occurrence of string subString begins.

start, if included, specifies the character position within srcString where the search begins. Default = 1 (the first character of srcString).

If srcString does not contain subString or start is > the length of srcString, returns zero.

inString("Hello there", "the")  ⇒ 7

"ABCEFG"→s1:If inString(s1, "D")=0:Disp "D not found."  ⇒ 0 not found.

int()  CATALOG

int(expression)  ⇒ integer
int(list)  ⇒ list
int(matrix)  ⇒ matrix

Returns the greatest integer that is less than or equal to the argument. This function is identical to floor().

The argument can be a real or a complex number.

For a list or matrix, returns the greatest integer of each of the elements.

int(ë 2.5)  ⇒ ë 3.

int([-1.234, 0, 0.37])  ⇒ [-2. 0 0.]

intDiv()  CATALOG

intDiv(number1, number2)  ⇒ integer
intDiv(list1, list2)  ⇒ list
intDiv(matrix1, matrix2)  ⇒ matrix

Returns the signed integer part of argument 1 divided by argument 2.

For lists and matrices returns the signed integer part of argument 1 divided by argument 2 for each element pair.

intDiv(ë 7, 2)  ⇒ ë 3

intDiv(4, 5)  ⇒ 0

intDiv({12, ë 14, ë 16}, {5, 4, ë 3})  ⇒ {2 ë 3 5}

integrate  See [], page 907.

iPart()  MATH/Number menu

iPart(number)  ⇒ integer
iPart(list)  ⇒ list
iPart(matrix)  ⇒ matrix

Returns the integer part of the argument.

For lists and matrices, returns the integer part of each element.

The argument can be a real or a complex number.

iPart(1.234)  ⇒ -1.

iPart({3/2, ë 2.37.003})  ⇒ {1 ë 2. 7.}

isArchiv()  CATALOG

isArchiv(var_name)  ⇒ true, false

Determines if var_name is archived or not. Returns true if var_name is archived. Returns false if var_name is not archived.

isArchiv(PROG1)  ⇒ True

isClkOn()  CATALOG

isClkOn()  ⇒ true, false

Determines if the clock is ON or OFF. Returns true if the clock is ON. Returns false if the clock is OFF.
### isLocked() CATALOG

**isLocked(var_name)** → true, false

- Determines if `var_name` is locked or not.
- Returns true if `var_name` is locked or archived.
- Returns false if `var_name` is not locked or archived.

### isPrime() MATH/Test menu

**isPrime(number)** ⇒ Boolean constant expression

- Returns true or false to indicate if `number` is a whole number > 2 that is evenly divisible only by itself and 1.
- If `number` exceeds about 306 digits and has no factors ≤ 1021, `isPrime(number)` displays an error message.
- If you merely want to determine if `number` is prime, use `isPrime()` instead of `factor()`. It is much faster, particularly if `number` is not prime and has a second-largest factor that exceeds about five digits.

### isVar() CATALOG

**isVar(var_name)** → true, false

- Determines if `var_name` is in use. Returns true if `var_name` exists. Returns false if `var_name` does not exist.

### Item CATALOG

- **Item itemNameString**, label
  - Valid only within a `Custom...EndCustm` or `ToolBar...EndTBar` block. Sets up a drop-down menu element to let you paste text to the cursor position (`Custom`) or branch to a label (`ToolBar`).
  - **Note:** Branching to a label is not allowed within a `Custom` block.

### Lbl CATALOG

**Lbl labelName**

- Defines a label with the name `labelName` in the program.
- You can use a `Goto labelName` instruction to transfer program control to the instruction immediately following the label.
  - `labelName` must meet the same naming requirements as a variable name.
**lcm()**  
**MATH/Number menu**

\[
lcm(\text{number1, number2}) \implies \text{expression}
\]

\[
lcm(\text{list1, list2}) \implies \text{list}
\]

\[
lcm(\text{matrix1, matrix2}) \implies \text{matrix}
\]

Returns the least common multiple of the two arguments. The \( \text{lcm} \) of two fractions is the \( \text{lcm} \) of their numerators divided by the \( \text{gcd} \) of their denominators. The \( \text{lcm} \) of fractional floating-point numbers is their product.

For two lists or matrices, returns the least common multiples of the corresponding elements.

\[
lcm(6,9) \implies 18
\]

\[
lcm(\{1/3, 14, 16\}, \{2/15, 7, 5\}) \implies \{2/3, 14, 80\}
\]

**left()**  
**MATH/String menu**

\[
\text{left(sourceString, num)} \implies \text{string}
\]

Returns the leftmost \( \text{num} \) characters contained in character string \( \text{sourceString} \).

If you omit \( \text{num} \), returns all of \( \text{sourceString} \).

\[
\text{left("Hello",2)} \implies \text{"He"}
\]

\[
\text{left(\{1,3,2,4\},3)} \implies \{1, 3, 2\}
\]

\[
\text{left(x<3)} \implies x
\]

**limit()**  
**MATH/Calculus menu**

\[
\text{limit(expression1, var, point, direction)} \implies \text{expression}
\]

\[
\text{limit(list1, var, point, direction)} \implies \text{list}
\]

\[
\text{limit(matrix1, var, point, direction)} \implies \text{matrix}
\]

Returns the limit requested.

- \( \text{direction} \): negative=from left, positive=from right, otherwise=both. (If omitted, \( \text{direction} \) defaults to both.)
- Limits at positive \( \infty \) and at negative \( \infty \) are always converted to one-sided limits from the finite side.

Depending on the circumstances, \( \text{limit()} \) returns itself or \( \text{undef} \) when it cannot determine a unique limit. This does not necessarily mean that a unique limit does not exist. \( \text{undef} \) means that the result is either an unknown number with finite or infinite magnitude, or it is the entire set of such numbers.

\[
\text{limit}(2x^3, x, 5)} \implies 13
\]

\[
\text{limit}(1/x, x, 0.1)} \implies \infty
\]

\[
\text{limit}(\sin(x)/x, x, 0)} \implies 1
\]

\[
\text{limit}((\sin(x+h)-\sin(x))/h, h, 0)} \implies \cos(x)
\]

\[
\text{limit}((\sin(1/n)\cdot n, n, \infty)} \implies e
\]

834  
Appendix A: Functions and Instructions
limit() uses methods such as L'Hôpital's rule, so there are unique limits that it cannot determine. If
expression1 contains undefined variables other than var, you might have to constrain them to
obtain a more concise result.

Limits can be very sensitive to rounding error. When possible, avoid the APPROX setting of the
Exact/Approx mode and approximate numbers when computing limits. Otherwise, limits that
should be zero or have infinite magnitude probably will not, and limits that should have
finite non-zero magnitude might not.

<table>
<thead>
<tr>
<th>limit(a^x,x,undef) ENTER</th>
<th>undef</th>
</tr>
</thead>
<tbody>
<tr>
<td>limit(a^x,x,a&gt;1) ENTER</td>
<td>∞</td>
</tr>
<tr>
<td>limit(a^x,x,a&gt;0 and a&lt;1)</td>
<td>0</td>
</tr>
</tbody>
</table>

**Line**

**CATALOG**

**Line xStart, yStart, xEnd, yEnd, drawMode**

Displays the Graph screen and draws, erases, or
inverts a line segment between the window
coordinates (xStart, yStart) and (xEnd, yEnd),
including both endpoints.

If drawMode = 1, draws the line (default).
If drawMode = 0, turns off the line.
If drawMode = -1, turns a line that is on to off or
off to on (inverts pixels along the line).

**Note:** Regraphing erases all drawn items. See
also PxlLine.

In the ZoomStd window, draw a line and then
erase it.

**Line 0,0,6,9 ENTER**

![Line drawing](image)

**LineHorz**

**CATALOG**

**LineHorz y[, drawMode]**

Displays the Graph screen and draws, erases, or
inverts a horizontal line at window position y.

If drawMode = 1, draws the line (default).
If drawMode = 0, turns off the line.
If drawMode = -1, turns a line that is on to off or
off to on (inverts pixels along the line).

**Note:** Regraphing erases all drawn items. See
also PxlHorz.

In a ZoomStd window:

**LineHorz 2.5 ENTER**

![LineHorz drawing](image)

Appendix A: Functions and Instructions
**LineTan**

**CATALOG**

**LineTan** \( \text{expression1, expression2} \)

Displays the Graph screen and draws a line tangent to \( \text{expression1} \) at the point specified.

\( \text{expression1} \) is an expression or the name of a function, where \( x \) is assumed to be the independent variable, and \( \text{expression2} \) is the \( x \) value of the point that is tangent.

**Note:** In the example shown, \( \text{expression1} \) is graphed separately. **LineTan** does not graph \( \text{expression1} \).

In function graphing mode and a ZoomTrig window:

\[
\text{Graph } \cos(x)
\]

1. \( \text{LineTan} \) \( \cos(x), \pi/4 \)

![Graph of \( \cos(x) \) with tangent line](image)

**LineVert**

**CATALOG**

**LineVert** \( x, \text{drawMode} \)

Displays the Graph screen and draws, erases, or inverts a vertical line at window position \( x \).

If \( \text{drawMode} = 1 \), draws the line (default).
If \( \text{drawMode} = 0 \), turns off the line.
If \( \text{drawMode} = -1 \), turns a line that is on to off or off to on (inverts pixels along the line).

**Note:** Regraphing erases all drawn items. See also **PxlVert**.

In a ZoomStd window:

\[
\text{LineVert} \ -2.5 \quad \text{ENTER}
\]

![Vertical line at position -2.5](image)

**LinReg**

**MATH/Statistics/Regressions menu**

**LinReg** \( \text{list1, list2, [list3], list4, list5} \)

Calculates the linear regression and updates all the system statistics variables.

All the lists must have equal dimensions except for \( \text{list5} \).

- \( \text{list1} \) represents \( x \)list.
- \( \text{list2} \) represents \( y \)list.
- \( \text{list3} \) represents frequency.
- \( \text{list4} \) represents category codes.
- \( \text{list5} \) represents category include list.

**Note:** \( \text{list1} \) through \( \text{list4} \) must be a variable name or \( \text{c1-c99} \) (columns in the last data variable shown in the Data/Matrix Editor). \( \text{list5} \) does not have to be a variable name and cannot be \( \text{c1-c99} \).

In function graphing mode:

\[
\{0.1,2,3,4,5,6\} \quad \text{ENTER}
\]

\[
\{0.2,3,4,3,4,6\} \quad \text{ENTER}
\]

\[
\text{LinReg} \ L1,L2 \quad \text{ENTER}
\]

\[
\text{ShowStat} \quad \text{ENTER}
\]

![Linear regression graph](image)

**ShowStat**

**NEW**
### \( \Delta \text{list()} \)  MATH/List menu

\[ \text{list}(\text{list1}) \Rightarrow \text{list} \]

Returns a list containing the differences between consecutive elements in \( \text{list1} \). Each element of \( \text{list1} \) is subtracted from the next element of \( \text{list1} \). The resulting list is always one element shorter than the original \( \text{list1} \).

\( \text{list}() \) \( \{20, 30, 45, 70\} \) \( \{10, 15, 25\} \)

### \( \text{list} \text{mat()} \)  MATH/List menu

\[ \text{list} \text{mat}(\text{list}, \text{elementsPerRow}) \Rightarrow \text{matrix} \]

Returns a matrix filled row-by-row with the elements from \( \text{list} \).

\( \text{elementsPerRow} \), if included, specifies the number of elements per row. Default is the number of elements in \( \text{list} \) (one row).

If \( \text{list} \) does not fill the resulting matrix, zeros are added.

\( \text{list} \{44, 44\} \text{mat} \{1, 2, 3\} \) \( \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \)

\( \text{list} \{44, 44\} \text{mat} \{1, 2, 3, 4, 5\}, 2 \) \( \begin{bmatrix} 1 & 2 & \varepsilon_1 \varepsilon_2 \varepsilon_3 \varepsilon_4 \varepsilon_5 \end{bmatrix} \)

### \( \ln() \)  MATH/String menu

\[ \ln(\text{expression}) \Rightarrow \text{expression} \]

Causes the input expression to be converted to an expression containing only natural logs (\( \ln() \)).

\( \ln(\text{log}(\text{expression})) \ln(\text{expression}) \)

\( \ln(\text{list}) \Rightarrow \text{list} \)

Returns the natural logarithm of the argument.

For a list, returns the natural logarithms of the elements.

\( \ln(\{\log(2.0)\}) \ln(2.0) \) \( .693... \)

\( \ln(\{-3.1, 2.5\}) \) \( \text{Error: Non-real result} \)

If complex format mode is REAL:

\( \ln(\{-3.1, 2.5\}) \) \( \text{Error: Non-real result} \)

If complex format mode is RECTANGULAR:

\( \ln(\{-3.1, 2.5\}) \) \( \ln(3) + i \cdot 1.828... \ln(5) \)

### \( \ln() \) 26B [\( \sqrt{\cdot} \) ] key

\[ \ln(\text{expression}) \Rightarrow \text{expression} \]

\[ \ln(\text{list}) \Rightarrow \text{list} \]

Returns the natural logarithm of the argument.

For a list, returns the natural logarithms of the elements.

\( \ln(\{\log(2.0)\}) \ln(2.0) \) \( .693... \)

\( \ln(\{-3.1, 2.5\}) \) \( \text{Error: Non-real result} \)

If complex format mode is REAL:

\( \ln(\{-3.1, 2.5\}) \) \( \text{Error: Non-real result} \)

If complex format mode is RECTANGULAR:

\( \ln(\{-3.1, 2.5\}) \) \( \ln(3) + i \cdot 1.828... \ln(5) \)

\( \ln(\text{squareMatrix}) \Rightarrow \text{squareMatrix} \)

Returns the matrix natural logarithm of \( \text{squareMatrix} \). This is not the same as calculating the natural logarithm of each element. For information about the calculation method, refer to \( \cos() \) on.

\( \text{squareMatrix} \) must be diagonalizable. The result always contains floating-point numbers.

\( \ln(\text{cos}()) \Rightarrow 1.831...+1.734...i \cdot .009...+1.490...i \cdot ... \)

\( \ln(\text{cos}()) \Rightarrow 1.831...+1.734...i \cdot .009...+1.490...i \cdot ... \)

In Radian angle mode and Rectangular complex format mode:

\( \ln(\{1.5, 3; 4, 2; 1, 6; -2, 1\}) \) \( \begin{bmatrix} 1.831...+1.734...i \cdot .009...+1.490...i \cdot ... \\ .448...+1.725...i \cdot 1.064...+1.790...i \cdot ... \\ .266...+2.083...i \cdot 1.124...+1.790...i \cdot ... \\ ... \end{bmatrix} \)

In Degree angle mode and Rectangular complex format mode:

\( \begin{bmatrix} 1.831...+1.734...i \cdot .009...+1.490...i \cdot ... \\ .448...+1.725...i \cdot 1.064...+1.790...i \cdot ... \\ .266...+2.083...i \cdot 1.124...+1.790...i \cdot ... \\ ... \end{bmatrix} \)

**Appendix A: Functions and Instructions** 

837
LnReg MATH/Statistics/Regressions menu

\[ \text{LnReg list1, list2[, (list3 [, list4, list5])]} \]

Calculates the logarithmic regression and updates all the system statistics variables.

All the lists must have equal dimensions except for list5.

list1 represents xlist.

list2 represents ylist.

list3 represents frequency.

list4 represents category codes.

list5 represents category include list.

Note: list1 through list4 must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). list5 does not have to be a variable name and cannot be c1–c99.

In function graphing mode:

\{1.2.3.4.5.6.7.8\} → L1 ENTER

\{1.2.2.3.3.3.4.4\} → L2 ENTER

LnReg L1,L2 ENTER Done

ShowStat ENTER

Regeq(x) "y1(x)" ENTER Done

NewPlot 1,1,L1,L2 ENTER Done

Local CATALOG

Local var1[, var2[, ...]]

Declares the specified vars as local variables. Those variables exist only during evaluation of a program or function and are deleted when the program or function finishes execution.

Note: Local variables save memory because they only exist temporarily. Also, they do not disturb any existing global variable values. Local variables must be used for For loops and for temporarily saving values in a multi-line function since modifications on global variables are not allowed in a function.

Lock CATALOG

Lock var1[, var2[, ...]]

Locks the specified variables. This prevents you from accidentally deleting or changing the variable without first using the unlock instruction on that variable.

In the example to the right, the variable L1 is locked and cannot be deleted or modified.

Note: The variables can be unlocked using the Unlock command.
**log()**

`log(expression1[,expression2]) ⇒ expression`

`log(list1[,expression2]) ⇒ list`

**Returns the base-expression2 logarithm of the argument.**

For a list, returns the base-expression2 logarithm of the elements.

If expression 2 is omitted, 10 is used as base.

**Example:**

```
log(2.0)
```

```
.301...
```

**If complex format mode is REAL:**

```
log({3,1.2,5})
```

Error: Non-real result

**If complex format mode is RECTANGULAR:**

```
log({3,1.2,5})
```

```
{log(3)+ 1.364…(i  .079… log(5)}
```

**log(squareMatrix1) ⇒ squareMatrix**

Returns the matrix base-expression2 logarithm of `squareMatrix1`. This is not the same as calculating the base-expression2 logarithm of each element. For information about the calculation method, refer to `cos()`.

`squareMatrix1` must be diagonalizable. The result always contains floating-point numbers.

**In Radian angle mode and Rectangular complex format mode:**

```
log([1,5,3;4,2,1;6,ë 2,1])
```

```
[.795…+.753…ø .003…- .647…i  …]
 [.194…-.315…i  .462…+.270…i  …]
[.115…-.904…i  .488…+.777…i  …]
```

**log(x,b) ⇒ expression**

`log(squareMatrix1) ⇒ squareMatrix`

In a list, returns the base expression 2 logarithm of the elements.

**Example:**

```
log(10,3) - log(5,3)
```

```
Log(10.3) - Log(5.3)
```

```
Log(2.0,4)
```

```
.5
```

**logbase**

**MATH/String menu**

`expression logbase(expression1) ⇒ expression`

Causes the input expression to be simplified to an expression using base `expression1`.

**Example:**

```
logbase(5)
```

```
Log(10.3) - Log(5.5) logbase(5)
```

```
Log(10.3) - Log(5.5)
```

```
log s(30)
```

```
log s(30)
```
Logistic \[ \text{MATH/Statistics/Regressions menu} \]

\[ \text{Logistic } \text{list1, list2[, [iterations], [list3], list4, list5]} \]

Calculates the logistic regression and updates all the system statistics variables.

- All the lists must have equal dimensions except for \text{list5}.
- \text{list1} represents \text{xlist}.
- \text{list2} represents \text{ylist}.
- \text{list3} represents frequency.
- \text{list4} represents category codes.
- \text{list5} represents category include list.

\text{iterations} specifies the maximum number of times a solution will be attempted. If omitted, 64 is used. Typically, larger values result in better accuracy but longer execution times, and vice versa.

\textbf{Note:} \text{list1} through \text{list4} must be a variable name or \text{c1–c99} (columns in the last data variable shown in the Data/Matrix Editor). \text{list5} does not have to be a variable name and cannot be \text{c1–c99}.

In function graphing mode:

\{1.2.3.4.5.6\}→L1 \[ \text{ENTER} \]
\{1.1.3.2.5.3.5.4.5.5.4.8\}→L2 \[ \text{ENTER} \]
\{1 1.3 2.5 ..\} →L3 
\text{Logistic L1,L2} \[ \text{ENTER} \] \text{Done}

\text{ShowStat} \[ \text{ENTER} \]

\text{Loop} \[ \text{CATALOG} \]

\text{Loop} \[ \text{Repeaterly executes the statements in } \text{block}. \text{Note that the loop will be executed endlessly, unless a } \text{Goto or Exit} \text{ instruction is executed within } \text{block}. \]

\text{block} is a sequence of statements separated with the \text{";"} character.

\textbf{Program segment:}

\text{:} \[ :i \rightarrow 1 \]
\text{:Loop} \[ :\text{Rand(6)} \rightarrow \text{die1} \]
\text{:Rand(6)} \rightarrow \text{die2} \[ :\text{If die1}=6 \text{ and die2}=6 \]
\text{: Goto End} \[ :i+1 \rightarrow i \]
\text{:EndLoop} \[ :\text{Lbl End} \]
\text{:Disp "The number of rolls is", i \[ :\]}

\text{840 Appendix A: Functions and Instructions}
Appendix A: Functions and Instructions

LU
MATH/Matrix menu

\[ \text{LU matrix, } \text{lMatName, uMatName, pMatName, tol} \]

Calculates the Doolittle LU (lower-upper) decomposition of a real or complex matrix. The lower triangular matrix is stored in \text{lMatName}, the upper triangular matrix in \text{uMatName}, and the permutation matrix (which describes the row swaps done during the calculation) in \text{pMatName}.

\[ \text{lMatName} \times \text{uMatName} = \text{pMatName} \times \text{matrix} \]

Optionally, any matrix element is treated as zero if its absolute value is less than \text{tol}. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, \text{tol} is ignored.

- If you use [ENTER] or set the mode to Exact/Approx=APPROXIMATE, computations are done using floating-point arithmetic.
- If \text{tol} is omitted or not used, the default tolerance is calculated as:
  \[ 5 \times 10^{-14} \times \max(\text{dim(matrix)}) \times \text{rowNorm(matrix)} \]

The LU factorization algorithm uses partial pivoting with row interchanges.

mat\text{data}
MATH/List menu

\[ \text{mat\text{data}, row1][col1][row2][col2]} \]

Converts a matrix to data. Each argument \text{row1}, \text{col1}, \text{row2}, \text{col2} can be individually omitted. If \text{row1} is omitted the default is 1. If \text{col1} is omitted the default is 1. If \text{row2} is omitted, the default is "max row." If \text{col2} is omitted, the default is "max column."
**matlist()**  MATH/List menu

\[
\text{matlist(matrix)} \Rightarrow \text{list}
\]

Returns a list filled with the elements in matrix. The elements are copied from matrix row by row.

\[
\text{matlist([1,2,3]) \rightarrow \{1,2,3\}}
\]

\[
\text{matlist([1,2,3;4,5,6]) \rightarrow \{1,2,3,4,5,6\}}
\]

**max()**  MATH/List menu

\[
\text{max(expression1, expression2)} \Rightarrow \text{expression}
\]

\[
\text{max(list1, list2)} \Rightarrow \text{list}
\]

\[
\text{max(matrix1, matrix2)} \Rightarrow \text{matrix}
\]

Returns the maximum of the two arguments. If the arguments are two lists or matrices, returns a list or matrix containing the maximum value of each pair of corresponding elements.

\[
\text{max(2.3,1.4)} \rightarrow 2.3
\]

\[
\text{max({1,2},{4,3})} \rightarrow \{1,3\}
\]

\[
\text{max(list)} \Rightarrow \text{expression}
\]

Returns the maximum element in list.

\[
\text{max({0,1,.3,.5}) \rightarrow 0.3}
\]

\[
\text{max(matrix)} \Rightarrow \text{matrix}
\]

Returns a row vector containing the maximum element of each column in matrix.

\[
\text{max([1,3,7;1,3,.4], list)} \rightarrow \{1,0,7\}
\]

**mean()**  MATH/Statistics menu

\[
\text{mean(list, freqlist)} \Rightarrow \text{expression}
\]

\[
\text{mean(matrix, freqmatrix)} \Rightarrow \text{matrix}
\]

Returns the mean of the elements in list. Each freqlist element counts the number of consecutive occurrences of the corresponding element in list.

\[
\text{mean({1,2,3,.4}) \rightarrow 2.6}
\]

\[
\text{mean({0,1,.3,.5}) \rightarrow 0.5}
\]

\[
\text{mean([1,3,7;1,3,.4], matrix)} \rightarrow \{1,0,7\}
\]

**median()**  MATH/Statistics menu

\[
\text{median(list)} \Rightarrow \text{expression}
\]

\[
\text{median(matrix)} \Rightarrow \text{matrix}
\]

Returns the median of the elements in list. Each freqlist element counts the number of consecutive occurrences of the corresponding element in list.

\[
\text{median({1,2,3,.4}) \rightarrow 0.2}
\]

\[
\text{median([1,3,7;1,3,.4], matrix)} \rightarrow \{1,0,7\}
\]
Appendix A: Functions and Instructions

**MedMed**

MATH/Statistics/Regressions menu

```
MedMed list1, list2, [list3], [list4], list5
```

Calculates the median-median line and updates all the system statistics variables.

All the lists must have equal dimensions except for list5.

- `list1` represents xlist.
- `list2` represents ylist.
- `list3` represents frequency.
- `list4` represents category codes.
- `list5` represents category include list.

Note: `list1` through `list4` must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). `list5` does not have to be a variable name and cannot be c1–c99.

In function graphing mode:

```
{0,1,2,3,4,5,6} L1 ENTER {0 1 2 3 4 5 6} L2 ENTER
```

```
MedMed L1 L2 ENTER
ShowStat ENTER
```

```
Regeq(x) y1(x) ENTER Done
```

```
NewPlot 1,1,L1,L2 ENTER Done
```

**mid()**

MATH/String menu

```
mid(sourceString, start, count) ⇒ string
```

Returns `count` characters from character string `sourceString`, beginning with character number `start`.

If `count` is omitted or is greater than the dimension of `sourceString`, returns all characters from `sourceString`, beginning with character number `start`.

`count` must be ≥ 0. If `count` = 0, returns an empty string.

```
mid("Hello there",2) ENTER "ello there"
mid("Hello there",7,3) ENTER "the"
mid("Hello there",1,5) ENTER "Hello"
mid("Hello there",1,0) ENTER ""
```

```
mid(sourceList, start, count) ⇒ list
```

Returns `count` elements from `sourceList`, beginning with element number `start`.

If `count` is omitted or is greater than the dimension of `sourceList`, returns all elements from `sourceList`, beginning with element number `start`.

`count` must be ≥ 0. If `count` = 0, returns an empty list.

```
mid({9,8,7,6},3) ENTER {7 6}
mid({9,8,7,6},2,2) ENTER {8 7}
mid({9,8,7,6},1,2) ENTER {9 8}
mid({9,8,7,6},1,0) ENTER {}
```

```
mid(sourceStringList, start, count) ⇒ list
```

Returns `count` strings from the list of strings `sourceStringList`, beginning with element number `start`.

```
mid("A","B","C","D",2,2) ENTER ("B" "C")
```
### min()  MATH/List menu

- **Syntax:**
  - `min(expression1, expression2)`
  - `min(list1, list2)`
  - `min(matrix1, matrix2)`

- **Description:**
  - Returns the minimum of the two arguments. If the arguments are two lists or matrices, returns a list or matrix containing the minimum value of each pair of corresponding elements.

```
min(2.3, 1.4) \rightarrow 1.4
min({0, 1, 3, .5}) \rightarrow -7
min([1, 2], [-4, 3]) \rightarrow [-4, 2]
```

### mod()  MATH/Number menu

- **Syntax:**
  - `mod(expression1, expression2)`
  - `mod(list1, list2)`
  - `mod(matrix1, matrix2)`

- **Description:**
  - Returns the first argument modulo the second argument as defined by the identities:
    - `mod(x, 0) \rightarrow x`
    - `mod(x, y) \rightarrow x - y \text{floor}(x/y)`
  - If the second argument is non-zero, the result is periodic in that argument. The result is either zero or has the same sign as the second argument.
  - If the arguments are two lists or two matrices, returns a list or matrix containing the modulo of each pair of corresponding elements.

```
mod(7, 0) \rightarrow 7
mod(7, 3) \rightarrow 1
mod(-7, 3) \rightarrow 2
mod(-7, -3) \rightarrow -2
mod(-7, 3) \rightarrow -1
mod(12, 14, 16, 9, 7, 5) \rightarrow [3, 0, 4]
```

### MoveVar  CATALOG

- **Syntax:**
  - `MoveVar var, oldFolder, newFolder`

- **Description:**
  - Moves variable `var` from `oldFolder` to `newFolder`. If `newFolder` does not exist, `MoveVar` creates it.

```
{1,2,3,4} \rightarrow L1 \rightarrow \text{Done}
MoveVar L1.Main.Games \rightarrow \text{Done}
```

### mRow()  MATH/Matrix/Row ops menu

- **Syntax:**
  - `mRow(expression, matrix1, index)`

- **Description:**
  - Returns a copy of `matrix1` with each element in `row index` of `matrix1` multiplied by `expression`.

```
mRow(-1/3, [1,2;3,4], 2) \rightarrow [-1, \frac{2}{3}]
```

### mRowAdd()  MATH/Matrix/Row ops menu

- **Syntax:**
  - `mRowAdd(expression, matrix1, index1, index2)`

- **Description:**
  - Returns a copy of `matrix1` with each element in `row index1` of `matrix1` replaced with:
    - `expression \times \text{row index1} + \text{row index2}`
### Appendix A: Functions and Instructions

#### nCr()

**MATH/Probability menu**

**nCr(expression1, expression2)** \(\Rightarrow\) **expression**

For integer `expression1` and `expression2` with `expression1 \geq expression2 \geq 0`, `nCr()` is the number of combinations of `expression1` things taken `expression2` at a time. (This is also known as a binomial coefficient.) Both arguments can be integers or symbolic expressions.

- **nCr(expression, 0)** \(\Rightarrow\) 1
- **nCr(expression, negInteger)** \(\Rightarrow\) 0
- **nCr(expression, posInteger)** \(\Rightarrow\) \(\frac{expression \cdot (expression-1) \cdot ... \cdot (expression-posInteger+1)}{posInteger!}\)
- **nCr(expression, nonInteger)** \(\Rightarrow\) \(\frac{expression!}{(expression-integer)! \cdot nonInteger!}\)

- **Example:**
  - `nCr(z,3)` \(\Rightarrow\) \(\frac{z \cdot (z-2) \cdot (z-1)}{6}\)
  - `nCr(z,c)` \(\Rightarrow\) \(\frac{z!}{c!(z-c)!}\)

### nDeriv()

**MATH/Calculus menu**

**nDeriv(expression1, var[, h])** \(\Rightarrow\) **expression**

Returns the numerical derivative as an expression. Uses the central difference quotient formula.

- **h** is the step value. If **h** is omitted, it defaults to 0.001.
- **Note:** See also `avgRC()` and `d()`.

- **Example:**
  - `nDeriv(cos(x),x,h)` \(\Rightarrow\) `limit(nDeriv(cos(x),x,h),h,0)`
  - `nDeriv(x^3,x,0.01)` \(\Rightarrow\) \(3 \cdot (x^{2} + .000033)\)
  - `nDeriv(cos(x),x)|x=\pi/2` \(\Rightarrow\) \(-1.\)
  - `nDeriv(x^2,x,{.01,.1})` \(\Rightarrow\) \(\{2 \cdot x \, 2 \cdot x\}\)

### NewData

**CATALOG**

**NewData dataVar, list1[, list2, ...]**

Creates data variable `dataVar`, where the columns are the lists in order.

- **Must have at least one list.**
- **list1, list2, ...** can be lists as shown, expressions that resolve to lists, or list variable names.

`NewData` makes the new variable current in the Data/Matrix Editor.

- **Example:**
  - `NewData mydata,{1,2,3},{4,5,6}` \(\Rightarrow\) Done

`NewData dataVar, matrix` Creates data variable `dataVar` based on `matrix`. 
**NewData sysData, matrix**

Loads the contents of `matrix` into the system data variable `sysData`.

**NewFold**  
**CATALOG**

NewFold `folderName`

Creates a user-defined folder with the name `folderName`, and then sets the current folder to that folder. After you execute this instruction, you are in the new folder.

**newList()**  
**CATALOG**

newList(numElements) ⇒ list

Returns a list with a dimension of `numElements`. Each element is zero.

**newMat()**  
**CATALOG also Math/Matrix menu**

newMat(numRows, numColumns) ⇒ matrix

Returns a matrix of zeros with the dimension `numRows` by `numColumns`.

**NewPic**  
**CATALOG**

NewPic `matrix, picVar[, maxRow[, maxCol]]`

Creates a pic variable `picVar` based on `matrix`. `matrix` must be an `n×2` matrix in which each row represents a pixel. Pixel coordinates start at 0,0. If `picVar` already exists, `NewPic` replaces it.

The default for `picVar` is the minimum area required for the matrix values. The optional arguments, `maxRow` and `maxCol`, determine the maximum boundary limits for `picVar`.

```
NewPic [1,1:2,2:3,3:4:4,5:5,1,4,2,4:1,5], xpic [ENTER] 
Done
```
**NewPlot**  CATALOG

`NewPlot n, type, xList[, yList[, frqList[, catList[, includeCatList[, mark[, bucketSize]]]]]]`

Creates a new plot definition for plot number `n`.

- `type` specifies the type of the graph plot.
  - 1 = scatter plot
  - 2 = xyline plot
  - 3 = box plot
  - 4 = histogram
  - 5 = modified box plot

- `mark` specifies the display type of the mark.
  - 1 =  (box)
  - 2 = × (cross)
  - 3 = + (plus)
  - 4 = ¤ (square)
  - 5 = ø (dot)

- `bucketSize` is the width of each histogram "bucket" (`type = 4`), and will vary based on the window variables `xmin` and `xmax`. `bucketSize` must be >0. Default = 1.

**Note:**
- `n` can be 1–9. Lists must be variable names or `c1–c99` (columns in the last data variable shown in the Data/Matrix Editor), except for `includeCatList`, which does not have to be a variable name and cannot be `c1–c99`.

---

**NewProb**  CATALOG

`NewProb`

Performs a variety of operations that let you begin a new problem from a cleared state without resetting the memory.

- Clears all single-character variable names (`Clear a–z`) in the current folder, unless the variables are locked or archived.
- Turns off all functions and stat plots (`FnOff` and `PlotsOff`) in the current graphing mode.
- Performs `ClrDraw`, `ClrErr`, `ClrGraph`, `ClrHome`, `ClrIO`, and `ClrTable`.

---

**nInt()**  MATH/Calculus menu

`nInt(expression, var, lower, upper)`

If the integrand `expression` contains no variable other than `var`, and if `lower` and `upper` are constants, positive `-`, or negative `+`, then `nInt()` returns an approximation of `{expression, var, lower, upper}`. This approximation is a weighted average of some sample values of the integrand in the interval `lower<var<upper`.

The goal is six significant digits. The adaptive algorithm terminates when it seems likely that the goal has been achieved, or when it seems unlikely that additional samples will yield a worthwhile improvement.

A warning is displayed ("Questionable accuracy") when it seems that the goal has not been achieved.
Nest \texttt{nInt()} to do multiple numeric integration. Integration limits can depend on integration variables outside them.

\begin{align*}
\text{nInt(nInt(} \int e^{x+y}/\sqrt{x^2-y^2}, \ y,-x,x,0.1) \ \text{ENTER} & \quad 3.304.. \\
\end{align*}

Note: See also \texttt{\textcolor{red}{\textbf{\textbackslash{}}}}.

\begin{center}
\textbf{norm()} \quad \textbf{MATH/Matrix/Norms menu}\end{center}

\begin{itemize}
\item \texttt{norm(matrix)} \Rightarrow expression
\item Returns the Frobenius norm.
\item \texttt{norm([a,b;c,d]) ENTER} \quad \sqrt{a^2+b^2+c^2+d^2}
\item \texttt{norm([1,2;3,4]) ENTER} \quad \sqrt{30}
\end{itemize}

\begin{center}
\textbf{not} \quad \textbf{MATH/Test menu}\end{center}

\begin{itemize}
\item \texttt{not Boolean expression1} \Rightarrow Boolean expression
\item Returns true, false, or a simplified Boolean expression.
\item \texttt{not 2>=3} \quad \texttt{true}
\item \texttt{not x<2} \quad x \geq 2
\item \texttt{not not innocent} \quad innocent
\item \texttt{not integer1} \Rightarrow integer
\item Returns the one's complement of a real integer. Internally, \texttt{integer1} is converted to a signed, 32-bit binary number. The value of each bit is flipped (0 becomes 1, and vice versa) for the one's complement. Results are displayed according to the Base mode.
\item You can enter the integer in any number base. For a binary or hexadecimal entry, you must use the \texttt{0b} or \texttt{0h} prefix, respectively. Without a prefix, the integer is treated as decimal (base 10).
\item If you enter a decimal integer that is too large for a signed, 32-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range.
\item \texttt{In Hex base mode:}
\item \texttt{not 0h7AC36 ENTER} \quad 0hFFF853C9
\item \textbf{Important}: Zero, not the letter O.
\item \texttt{In Bin base mode:}
\item \texttt{0b100101 ENTER} \quad 37
\item \texttt{not 0b100101 ENTER} \quad 0b11111111111111111111111111011010
\item \texttt{ans(1) ENTER} \quad -38
\item \textbf{Note}: A binary entry can have up to 32 digits (not counting the \texttt{0b} prefix). A hexadecimal entry can have up to 8 digits.
\item \textbf{Note}: To type the \textbf{\textcolor{red}{\textbackslash{}}} conversion operator, press \texttt{2 	extbackslash{}}. You can also select base conversions from the \textbf{MATH/Base menu}.
\end{itemize}

\begin{center}
\textbf{nPr()} \quad \textbf{MATH/Probability menu}\end{center}

\begin{itemize}
\item \texttt{nPr(expression1, expression2)} \Rightarrow expression
\item For integer \texttt{expression1} and \texttt{expression2} with \texttt{expression1} \geq \texttt{expression2} \geq 0, \texttt{nPr()} is the number of permutations of \texttt{expression1} things taken \texttt{expression2} at a time. Both arguments can be integers or symbolic expressions.
\item \texttt{nPr(5,3) ENTER} \quad \texttt{5!/(5-3)!}
\item \texttt{ans(1) ENTER} \quad 120
\item \texttt{nPr(3,0) ENTER} \quad 1
\item \texttt{nPr(expression, negInteger)} \Rightarrow \texttt{1/((expression+1) \ldots (expression+negInteger))}
\item \texttt{nPr(expression, posInteger)} \Rightarrow \texttt{expression \ldots (expression-posInteger+1)}
\item \texttt{nPr(expression, nonInteger)} \Rightarrow \texttt{expression!/(expression-nonInteger)!}
\item \texttt{nPr(list1, list2)} \Rightarrow list
\item Returns a list of permutations based on the corresponding element pairs in the two lists. The arguments must be the same size list.
\item \texttt{nPr([5,4,3],[2,4,2]) ENTER} \quad \texttt{20 24 6}
\end{itemize}
\[ nPr(\text{matrix}_1, \text{matrix}_2) \Rightarrow \text{matrix} \]

Returns a matrix of permutations based on the corresponding element pairs in the two matrices. The arguments must be the same size matrix.

**nSolve() MATH/Algebra menu**

\[ nSolve(\text{equation}, \text{varOrGuess}) \Rightarrow \text{number or error_string} \]

Iteratively searches for one approximate real numeric solution to \( \text{equation} \) for its one variable. Specify \( \text{varOrGuess} \) as:

- variable
- or
- variable = real number

For example, \( x = 3 \) is valid and so is \( x = 3 \).

\( nSolve() \) is often much faster than \( \text{solve()} \) or \( \text{zeros()} \), particularly if the "|" operator is used to constrain the search to a small interval containing exactly one simple solution.

\( nSolve() \) attempts to determine either one point where the residual is zero or two relatively close points where the residual has opposite signs and the magnitude of the residual is not excessive. If it cannot achieve this using a modest number of sample points, it returns the string "no solution found."

If you use \( nSolve() \) in a program, you can use \( \text{getType()} \) to check for a numeric result before using it in an algebraic expression.

**Note:** See also \( \text{cSolve()}, \text{cZeros()}, \text{solve()}, \) and \( \text{zeros()} \).

**OneVar MATH/Statistics menu**

\[ \text{OneVar list1}, \text{list2}, \text{list3}, \text{list4} \]

Calculates 1-variable statistics and updates all the system statistics variables.

All the lists must have equal dimensions except for \( \text{list2} \).

\( \text{list1} \) represents \( x \)-list.

\( \text{list2} \) represents frequency.

\( \text{list3} \) represents category codes.

\( \text{list4} \) represents category include list.

**Note:** \( \text{list1} \) through \( \text{list4} \) must be a variable name or \( 1-99 \) (columns in the last data variable shown in the Data/Matrix Editor). \( \text{list4} \) does not have to be a variable name and cannot be \( 1-99 \).
OR

MATH/Test menu

Boolean expression1 or Boolean expression2 ⇒ Boolean expression

Returns true or false or a simplified form of the original entry.

Returns true if either or both expressions simplify to true. Returns false only if both expressions evaluate to false.

Note: See xor.

\[ \text{integer1 or integer2} \Rightarrow \text{integer} \]

Compares two real integers bit-by-bit using an or operation. Internally, both integers are converted to signed, 32-bit binary numbers. When corresponding bits are compared, the result is 1 if either bit is 1; the result is 0 only if both bits are 0. The returned value represents the bit results, and is displayed according to the Base mode.

You can enter the integers in any number base. For a binary or hexadecimal entry, you must use the 0b or 0h prefix, respectively. Without a prefix, integers are treated as decimal (base 10).

If you enter a decimal integer that is too large for a signed, 32-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range.

Note: See xor.

ord() MATH/String menu

\[ \text{ord(string)} \Rightarrow \text{integer} \]
\[ \text{ord(list)} \Rightarrow \text{list} \]

Returns the numeric code of the first character in character string string, or a list of the first characters of each list element.

See Appendix B for a complete listing of character codes.

\[ \text{ord(“hello”)} \Rightarrow 104 \]
\[ \text{char(104)} \Rightarrow “h” \]
\[ \text{ord(char(24))} \Rightarrow 24 \]
\[ \text{ord(“alpha”, “beta”)} \Rightarrow \{97 98\} \]

Output CATALOG

\[ \text{Output row, column, exprOrString} \]

Displays exprOrString (an expression or character string) on the Program I/O screen at the text coordinates \( \text{row, column} \).

An expression can include conversion operations such as \( \text{DD} \) and \( \text{Rect} \). You can also use the \( \Rightarrow \) operator to perform unit and number base conversions.

If Pretty Print = ON, exprOrString is "pretty printed."

From the Program I/O screen, you can press \( \text{FE} \) to display the Home screen, or a program can use DispHome.

Program segment:

\[ \begin{align*}
\text{:RandSeed 1147} \\
\text{:ClrIO} \\
\text{:For 1,1.90,10} \\
\text{\quad \text{Output 1, rand(100), “Hello”}} \\
\text{:EndFor} \\
\text{\quad \text{:DispHome}}
\end{align*} \]

Result after execution:

```
Hello
Hello
Hello
Hello
Hello
```

Important: Zero, not the letter O.
### Appendix A: Functions and Instructions

#### Rx()

**MATH/Angle menu**

\[ \text{Rx}(r, q) \Rightarrow \text{expression} \]

Returns the equivalent x-coordinate of the \((r, q)\) pair.

**Note:** The \(q\) argument is interpreted as either a degree, gradian or radian angle, according to the current angle mode. If the argument is an expression, you can use \(\text{deg}\), \(\text{grad}\) or \(\text{rad}\) to override the angle mode setting temporarily.

In Radian angle mode:

\[ \text{Rx}(r, \text{deg}60) \Rightarrow 2 \]

\[ \text{Rx}(\{\text{deg}3, \text{deg}10, \text{deg}1.3\}, \{\text{deg}60, \text{deg}4.0\}) \Rightarrow \{ -3/2 \, 5\sqrt{2} \, 1.3 \} \]

#### Ry()

**MATH/Angle menu**

\[ \text{Ry}(r, q) \Rightarrow \text{expression} \]

Returns the equivalent y-coordinate of the \((r, q)\) pair.

**Note:** The \(q\) argument is interpreted as either a degree, radian or gradian angle, according to the current angle mode. If the argument is an expression, you can use \(\text{deg}\), \(\text{rad}\) or \(\text{grad}\) to override the angle mode setting temporarily.

In Radian angle mode:

\[ \text{Ry}(r, \text{deg}60) \Rightarrow 2 \cdot \sqrt{3} \]

\[ \text{Ry}(\{\text{deg}3, \text{deg}10, \text{deg}1.3\}, \{\text{deg}60, \text{deg}4.0\}) \Rightarrow \{ -\frac{3\cdot\sqrt{3}}{2} \, -5\sqrt{2} \, 0 \} \]

#### part()

**CATALOG**

\[ \text{part}(\text{expression}, \text{nonNegativeInteger}) \]

This advanced programming function lets you identify and extract all of the sub-expressions in the simplified result of \(\text{expression}\).

For example, if \(\text{expression1}\) simplifies to \(\cos(\pi \times x+3)\):

- The \(\cos()\) function has one argument: \((\pi \times x+3)\).
- The sum of \((\pi \times x+3)\) has two operands: \(\pi \times x\) and 3.
- The number 3 has no arguments or operands.
- The product \(\pi \times x\) has two operands: \(\pi\) and \(x\).
- The variable \(x\) and the symbolic constant \(\pi\) have no arguments or operands.

If \(x\) has a numeric value and you press \(\text{CF} \Rightarrow\), the numeric value of \(\pi \times x\) is calculated, the result is added to 3, and then the cosine is calculated. \(\cos()\) is the **top-level** operator because it is applied last.

\[ \text{part}(\text{expression}) \Rightarrow \text{number} \]

Simplifies \(\text{expression}\) and returns the number of top-level arguments or operands. This returns 0 if \(\text{expression}\) is a number, variable, or symbolic constant such as \(\pi\), 0, \(\text{e}\), or \(\text{i}\).

\[ \text{part}(\text{expression}, 0) \Rightarrow \text{string} \]

Simplifies \(\text{expression}\) and returns a string that contains the top-level function name or operator. This returns \(\text{string}\) if \(\text{expression}\) is a number, variable, or symbolic constant such as \(\pi\), 0, \(\text{e}\), or \(\text{i}\).

\[ \text{part}(\cos(\pi \times x+3)) \Rightarrow 1 \]

**Note:** \(\cos(\pi \times x+3)\) has one argument.

\[ \text{part}(\cos(\pi \times x+3), 0) \Rightarrow \text{"cos"} \]
part(expression, n) ⇒ expression

Simplifies expression and returns the n argument or operand, where n is > 0 and ≤ the number of top-level arguments or operands returned by part(expression). Otherwise, an error is returned.

By combining the variations of part(), you can extract all of the sub-expressions in the simplified result of expression. As shown in the example to the right, you can store an argument or operand and then use part() to extract further sub-expressions.

Note: When using part(), do not rely on any particular order in sums and products.

Expressions such as (x+y+z) and (x−y−z) are represented internally as (x+y)+z and (x−y)]. This affects the values returned for the first and second argument. There are technical reasons why part(x+y+z,1) returns y+x instead of x+y.

Similarly, x y z is represented internally as (x y z). Again, there are technical reasons why the first argument is returned as y x instead of x y.

When you extract sub-expressions from a matrix, remember that matrices are stored as lists of lists, as illustrated in the example to the right.

The example Program Editor function to the right uses getType() and part() to partially implement symbolic differentiation. Studying and completing this function can help teach you how to differentiate manually. You could even include functions that the cannot differentiate, such as Bessel functions.

part(cos(x•x+3),1) ENTER 3•x x

part(cos(x•x+3)) ENTER 1
part(cos(x•x+3),0) ENTER "cos"
part(cos(x•x+3),1) TEMP ENTER 3•x x

Note: Simplification changed the order of the argument.

part(cos(p θ x+3),1)¸ 3+p θ x

part(x+y+z) ENTER 2
part(x+y+z,2) ENTER z
part(x+y+z,1) ENTER y x

Expressions such as (x+y+z) and (x•y•z) are represented internally as (x+y)+z and (x•y•z). This affects the values returned for the first and second argument. There are technical reasons why part(x+y+z,1) returns y+x instead of x+y.

Similarly, x y z is represented internally as (x y z). Again, there are technical reasons why the first argument is returned as y x instead of x y.

When you extract sub-expressions from a matrix, remember that matrices are stored as lists of lists, as illustrated in the example to the right.

The example Program Editor function to the right uses getType() and part() to partially implement symbolic differentiation. Studying and completing this function can help teach you how to differentiate manually. You could even include functions that the cannot differentiate, such as Bessel functions.
### PassErr

**CATALOG**

**PassErr**

Passes an error to the next level.

If "errornum" is zero, **PassErr** does not do anything.

The **Else** clause in the program should use **ClrErr** or **PassErr**. If the error is to be processed or ignored, use **ClrErr**. If what to do with the error is not known, use **PassErr** to send it to the next error handler. (See also **ClrErr**.)

**See ClrErr program listing example.**

### Pause

**CATALOG**

**Pause**

Suspended program execution. If you include expression, displays expression on the Program I/O screen.

expression can include conversion operations such as **→DD** and **→Rect**. You can also use the > operator to perform unit and number base conversions.

If the result of expression is too big to fit on a single screen, you can use the cursor pad to scroll the display.

Program execution resumes when you press **ENTER**.

**Program segment:**

```plaintext
:ClrIO
:DelVar temp
:1 "temp[1]
:1 "temp[2]
:Disp temp[2]
:Guess the Pattern
:For 1,3,20
  :temp[i-2]+temp[i-1]-temp[i]
  :Disp temp[i]
  :Disp temp,"Can you guess the next","number?"
 :Pause
 :EndFor
```

### PlotsOff

**CATALOG**

**PlotsOff**

- **PlotsOff** 
- **PlotsOff ENTER**
- **Done**

Turns off the specified plots for graphing. When in 2-graph mode, only affects the active graph.

If no parameters, then turns off all plots.

### PlotsOn

**CATALOG**

**PlotsOn**

- **PlotsOn** 
- **PlotsOn ENTER**
- **Done**

Turns on the specified plots for graphing. When in 2-graph mode, only affects the active graph.

If no include any arguments, turns on all plots.

### Polar

**MATH/Matrix/Vector ops menu**

<table>
<thead>
<tr>
<th>Vector</th>
<th><strong>Polar</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>→Polar</strong></td>
<td><strong>→Polar</strong></td>
</tr>
</tbody>
</table>

Displays vector in polar form \([r \angle \theta]\). The vector must be of dimension 2 and can be a row or a column.

**Note:** **→Polar** is a display-format instruction, not a conversion function. You can use it only at the end of an entry line, and it does not update ans.

**Note:** See also **→Rect**.
complexValue \rightarrow \text{Polar}

Displays \text{complexVector} in polar form.

- Degree angle mode returns \((r \angle \theta)\).
- Radian angle mode returns \(re^{i\theta}\).

\text{complexValue} can have any complex form. However, an \(re^{i\theta}\) entry causes an error in Degree angle mode.

\textbf{Note:} You must use the parentheses for an \((r \angle \theta)\) polar entry.

\begin{align*}
3 \angle 3 \text{Polar} \Rightarrow e^{3/3} \\
(4 \angle \pi/3) \text{Polar} \Rightarrow e^{3/4}
\end{align*}

\begin{align*}
\text{In Radian angle mode:} \\
(4 \angle \pi/3) \text{Polar} \Rightarrow e^{3/4}
\end{align*}

\begin{align*}
\text{In Gradian angle mode:} \\
4 \angle 100 \Rightarrow e^{p3/4}
\end{align*}

\begin{align*}
\text{In Degree angle mode:} \\
3 \angle 3 \text{Polar} \Rightarrow (5 \angle 90 - \tan(3/4))
\end{align*}

\begin{tabular}{|l|}
\hline
\textbf{polyEval() MATH/List menu} \\
\hline
\text{polyEval}([list1], expression) \Rightarrow \text{expression} \\
\text{polyEval}([list1], [list2]) \Rightarrow \text{expression}
\hline
\end{tabular}

Interprets the first argument as the coefficient of a descending-degree polynomial, and returns the polynomial evaluated for the value of the second argument.

\begin{align*}
\text{polyEval}([1,2,3,4], x) \Rightarrow a \cdot x^3 + b \cdot x^2 + c
\end{align*}

\begin{align*}
\text{polyEval}([1,2,3,4], 2) \Rightarrow 26 \\
\text{polyEval}([1,2,3,4], [2, -7]) \Rightarrow \{26 - 262\}
\end{align*}

\begin{tabular}{|l|}
\hline
\textbf{PopUp CATALOG} \\
\hline
\text{PopUp} \quad \text{ItemList, var}
\hline
\end{tabular}

Displays a pop-up menu containing the character strings from \text{ItemList}, waits for you to select an item, and stores the number of your selection in \text{var}.

The elements of \text{ItemList} must be character strings: 

\{item1String, item2String, item3String, ...\}

If \text{var} already exists and has a valid item number, that item is displayed as the default choice.

\text{ItemList} must contain at least one choice.
Appendix A: Functions and Instructions

### PowerReg

**MATH/Statistics/Regressions menu**

PowerReg \(\text{list1, list2, list3, list4, list5}\)

Calculates the power regression and updates all the system statistics variables.

All the lists must have equal dimensions except for list5.

- list1 represents xlist.
- list2 represents ylist.
- list3 represents frequency.
- list4 represents category codes.
- list5 represents category include list.

**Note:** list1 through list4 must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). list5 does not have to be a variable name and cannot be c1–c99.

In function graphing mode:

1. \(\{1,2,3,4,5,6,7\} \rightarrow \text{L1}\) \(\text{ENTER}\)
2. \(\{1,2,3,4,6\} \rightarrow \text{L2}\) \(\text{ENTER}\)
3. \(\text{PowerReg}\ \text{L1}, \text{L2}\) \(\text{ENTER}\)
4. \(\text{ShowStat}\) \(\text{ENTER}\)

\(\text{PowerReg}\) \(\text{L1}, \text{L2}\)

**MATH/Statistics/Regressions menu**

### Prgm

**CATALOG**

Prgm

Program segment:

```
:prgmname()
:Prgm
:
:EndPrgm
```

**Prompt**

**CATALOG**

Prompt \(\text{var1, var2, ...}\)

Displays a prompt on the Program I/O screen for each variable in the argument list, using the prompt var1. Stores the entered expression in the corresponding variable.

**Prompt** must have at least one argument.

### Product (π)

**See Π(), page 908.**

**Product()**

**MATH/List menu**

product(\(\text{list1, start, end}\)) \(\Rightarrow\) \(\text{expression}\)

- Returns the product of the elements contained in list1. Start and end are optional. They specify a range of elements.

- \(\text{product}\{\{1,2,3,4\}\} \rightarrow \text{ENTER}\) \(24\)
- \(\text{product}\{\{2,x\,y\}\} \rightarrow \text{ENTER}\) \(2\times x\times y\)
- \(\text{product}\{\{4,5,8,9\}, 2, 3\} \rightarrow \text{ENTER}\) \(40\)

product(\(\text{matrix1, start, end}\)) \(\Rightarrow\) \(\text{matrix}\)

- Returns a row vector containing the products of the elements in the columns of matrix1. Start and end are optional. They specify a range of rows.

- \(\text{product}\{\{1,2,3;4,5,6;7,8,9\}\} \rightarrow \text{ENTER}\) \([28 \, 80 \, 162]\)
- \(\text{product}\{\{1,2;3;4,5;6;7,8,9\}, 1, 2\} \rightarrow \text{ENTER}\) \([4,10,18]\)
propFrac()  MATH/Algebra menu

\[ \text{propFrac}(\text{expression}, \text{var}) \Rightarrow \text{expression} \]

**propFrac(expression1, var)**

- **propFrac(rational, number)** returns **rational, number** as the sum of an integer and a fraction having the same sign and a greater denominator magnitude than numerator magnitude.

- **propFrac(rational, expression, var)** returns the sum of proper ratios and a polynomial with respect to **var**. The degree of **var** in the denominator exceeds the degree of **var** in the numerator in each proper ratio. Similar powers of **var** are collected. The terms and their factors are sorted with **var** as the main variable.

If **var** is omitted, a proper fraction expansion is done with respect to the most main variable. The coefficients of the polynomial part are then made proper with respect to their most main variable first and so on.

For rational expressions, **propFrac()** is a faster but less extreme alternative to **expand()**.

**propFrac(4/3)**  \[ \text{\(\Rightarrow 1 + \frac{1}{3}\)} \]

**propFrac(\(\frac{x^2+y+1}{y+1}\)).\(x\)**  \[ \text{\(\Rightarrow \text{propFrac(ans(1))}\)} \]

**propFrac(\((x^2+x+1)/(x+1)+(y^2+y+1)/(y+1)\),x)**  \[ \text{\(\Rightarrow \text{propFrac(ans(1))}\)} \]

PtChg  CATALOG

<table>
<thead>
<tr>
<th>PtChg (x, y)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PtChg (xList, yList)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Displays the Graph screen and reverses the screen pixel nearest to window coordinates \((x, y)\).

Note: PtChg through PtText show continuing similar examples.

PtChg 2,4 ENTER

PtOff  CATALOG

<table>
<thead>
<tr>
<th>PtOff (x, y)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PtOff (xList, yList)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Displays the Graph screen and turns off the screen pixel nearest to window coordinates \((x, y)\).

PtOff 2,4 ENTER

PtOn  CATALOG

<table>
<thead>
<tr>
<th>PtOn (x, y)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PtOn (xList, yList)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Displays the Graph screen and turns on the screen pixel nearest to window coordinates \((x, y)\).

PtOn 3,5 ENTER

ptTest()  CATALOG

<table>
<thead>
<tr>
<th>ptTest ((x, y)) \Rightarrow \text{Boolean constant expression}</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ptTest ((xList, yList)) \Rightarrow \text{Boolean constant expression}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns true or false. Returns true only if the screen pixel nearest to window coordinates \((x, y)\) is on.

ptTest(3,5) ENTER  \[ \text{true} \]
PtText function

\[
\text{PtText(string \ x, \ y)}
\]

Displays the Graph screen and places the character string \text{string} on the screen at the pixel nearest the specified \((x, y)\) window coordinates. \text{string} is positioned with the upper-left corner of its first character at the coordinates.

Example:

\[
\text{PtText("sample",3.5) \ \text{enter}}
\]

PxlChg function

\[
\text{PxlChg(row, \ col)}
\]

Displays the Graph screen and reverses the pixel at pixel coordinates \((row, \ col)\).

**Note:** Regraphing erases all drawn items.

Example:

\[
\text{PxlChg(2,4) \ \text{enter}}
\]

PxlCrcl function

\[
\text{PxlCrcl(row, \ col, \ r, \ drawMode)}
\]

Displays the Graph screen and draws a circle centered at pixel coordinates \((row, \ col)\) with a radius of \(r\) pixels.

- If \text{drawMode} = 1, draws the circle (default).
- If \text{drawMode} = 0, turns off the circle.
- If \text{drawMode} = -1, inverts pixels along the circle.

**Note:** Regraphing erases all drawn items. See also \text{Circle}.

Example:

\[
\text{PxlCrcl(40,80,30,1) \ \text{enter}}
\]

PxlHorz function

\[
\text{PxlHorz(row, \ drawMode)}
\]

Displays the Graph screen and draws a horizontal line at pixel position \text{row}.

- If \text{drawMode} = 1, draws the line (default).
- If \text{drawMode} = 0, turns off the line.
- If \text{drawMode} = -1, turns a line that is on to off or off to on (inverts pixels along the line).

**Note:** Regraphing erases all drawn items. See also \text{LineHorz}.

Example:

\[
\text{PxlHorz(25,1) \ \text{enter}}
\]

PxlLine function

\[
\text{PxlLine(rowStart, \ colStart, \ rowEnd, \ colEnd, \ drawMode)}
\]

Displays the Graph screen and draws a line between pixel coordinates \((rowStart, \ colStart)\) and \((rowEnd, \ colEnd)\), including both endpoints.

- If \text{drawMode} = 1, draws the line (default).
- If \text{drawMode} = 0, turns off the line.
- If \text{drawMode} = -1, turns a line that is on to off or off to on (inverts pixels along the line).

**Note:** Regraphing erases all drawn items. See also \text{Line}.

Example:

\[
\text{PxlLine(50,15,20,90,1) \ \text{enter}}
\]
PxlOff  CATALOG

PxlOff row, col
PxlOff rowList, colList

Displays the Graph screen and turns off the pixel at pixel coordinates (row, col).

Note: Regraphing erases all drawn items.

\[
\begin{array}{|c|c|c|c|}
\hline
& & & \\
\hline
\end{array}
\]

PxlOn  CATALOG

PxlOn row, col
PxlOn rowList, colList

Displays the Graph screen and turns on the pixel at pixel coordinates (row, col).

Note: Regraphing erases all drawn items.

\[
\begin{array}{|c|c|c|c|}
\hline
. & & & \\
\hline
\end{array}
\]

pxlTest()  CATALOG

pxlTest (row, col) ⇒ Boolean expression
pxlTest (rowList, colList) ⇒ Boolean expression

Returns true if the pixel at pixel coordinates (row, col) is on. Returns false if the pixel is off.

Note: Regraphing erases all drawn items.

\[
\begin{array}{|c|c|}
\hline
true & \\
\hline
false & \\
\end{array}
\]

PxlText  CATALOG

PxlText string, row, col

Displays the Graph screen and places character string on the screen, starting at pixel coordinates (row, col).

string is positioned with the upper-left corner of its first character at the coordinates.

Note: Regraphing erases all drawn items.

\[
\begin{array}{|c|c|c|c|}
\hline
& & & \\
\hline
\end{array}
\]

PxlVert  CATALOG

PxlVert col[, drawMode]

Draws a vertical line down the screen at pixel position col.

If drawMode = 1, draws the line (default).
If drawMode = 0, turns off the line.
If drawMode = -1, turns a line that is on to off or off to on (inverts pixels along the line).

Note: Regraphing erases all drawn items. See also LineVert.
**QR**

**MATH/Matrix menu**

Calculates the Householder QR factorization of a real or complex matrix. The resulting Q and R matrices are stored to the specified MatNames.

The Q matrix is unitary. The R matrix is upper triangular.

 Optionally, any matrix element is treated as zero if its absolute value is less than tol. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, tol is ignored.

- If you use \( \text{QuadReg} \) or set the mode to Exact/Approx=APPROXIMATE, computations are done using floating-point arithmetic.

- If tol is omitted or not used, the default tolerance is calculated as:

  \[
  5 \times 10^{-14} \cdot \max(\text{dim}(\text{matrix})) \cdot \text{rowNorm}(\text{matrix})
  \]

The QR factorization is computed numerically using Householder transformations. The symbolic solution is computed using Gram-Schmidt. The columns in qMatName are the orthonormal basis vectors that span the space defined by matrix.

**QuadReg**

**MATH/Statistics/Regressions menu**

Calculates the quadratic polynomial regression and updates the system statistics variables.

All the lists must have equal dimensions except for list5.

- list1 represents xlist.
- list2 represents ylist.
- list3 represents frequency.
- list4 represents category codes.
- list5 represents category include list.

In function graphing mode:

\[
\{0,1,2,3,4,5,6,7\} \text{ ENTER} \\
\{1 2 3 \ldots\} \\
\{4,3,1,1,2,2,3,3\} \text{ ENTER} \\
\{1 2 3 \ldots\} \\
\text{QuadReg L1,L2 ENTER} \\
\text{Done}
\]

ShowStat

\[
\{1 2 3 \ldots\} \\
\{4 3 1 \ldots\} \\
\text{QuadReg L1,L2 ENTER} \\
\text{Done}
\]

ShowStat

\[
\{1 2 3 \ldots\} \\
\{4 3 1 \ldots\} \\
\text{QuadReg L1,L2 ENTER} \\
\text{Done}
\]

\[
\{1 2 3 \ldots\} \\
\{4 3 1 \ldots\} \\
\text{QuadReg L1,L2 ENTER} \\
\text{Done}
\]

\[
\{1 2 3 \ldots\} \\
\{4 3 1 \ldots\} \\
\text{QuadReg L1,L2 ENTER} \\
\text{Done}
\]

\[
\{1 2 3 \ldots\} \\
\{4 3 1 \ldots\} \\
\text{QuadReg L1,L2 ENTER} \\
\text{Done}
\]

\[
\{1 2 3 \ldots\} \\
\{4 3 1 \ldots\} \\
\text{QuadReg L1,L2 ENTER} \\
\text{Done}
\]

**QR matrix, qMatName, rMatName, tol**

\[
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{bmatrix}
\]

\[
\text{QR m1.qm,rm ENTER Done}
\]

\[
\begin{bmatrix}
.123 & .904 & .408 \\
.492 & .301 & .816 \\
.861 & -.301 & .408
\end{bmatrix}
\]

\[
\text{qr ENTER}
\]

\[
\begin{bmatrix}
.124 & 9.601 & 11.078 \\
0 & .904 & 1.809 \\
0 & 0 & 0
\end{bmatrix}
\]

\[
\text{rm ENTER}
\]

\[
\begin{bmatrix}
\text{m,n,o,p}\|
\text{m1}\|
\text{qm}\|
\text{rm}
\end{bmatrix}
\]

\[
\begin{bmatrix}
\text{m,n,o,p}\|
\text{m1}\|
\text{qm}\|
\text{rm}
\end{bmatrix}
\]

The floating-point number (9.) in m1 causes results to be calculated in floating-point form.
Note: list1 through list4 must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). list5 does not have to be a variable name and cannot be c1–c99.

Regeq(x) = \( y_1(x) \)

Done

NewPlot 1,1,L1,L2 [GRAPH]

QuartReg MATH/Statistics/Regressions menu

QuartReg list1, list2, [list3, list4, list5]

Calculates the quartic polynomial regression and updates the system statistics variables.

All the lists must have equal dimensions except for list5.

list1 represents xlist.
list2 represents ylist.
list3 represents frequency.
list4 represents category codes.
list5 represents category include list.

Note: list1 through list4 must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). list5 does not have to be a variable name and cannot be c1–c99.

In function graphing mode:

\[ \{ -2, -1, 0, 1, 2, 3, 4, 5, 6 \} \] \[ L1 \] [ENTER]

\[ \begin{array}{c}
-2 \\
-1 \\
0 \\
1 \\
2 \\
3 \\
4 \\
5 \\
6
\end{array} \]

\[ \{ \quad \} \] \[ L2 \] [GRAPH]

QuartReg L1,L2 [GRAPH]

ShowStat [GRAPH]

In function graphing mode:

\[ \{ -2, -1, 0, 1, 2, 3, 4, 5, 6 \} \] \[ L1 \] [ENTER]

\[ \begin{array}{c}
-2 \\
-1 \\
0 \\
1 \\
2 \\
3 \\
4 \\
5 \\
6
\end{array} \]

\[ \{ \quad \} \] \[ L2 \] [GRAPH]

QuartReg L1,L2 [GRAPH]

ShowStat [GRAPH]
\text{R} \text{P0()} \quad \text{MATH/Angle menu}

\text{R} \text{P0()} \quad (\text{xExpression}, \text{yExpression}) \Rightarrow \text{expression}
\text{R} \text{P0()} \quad (\text{List1}, \text{List2}) \Rightarrow \text{list}
\text{R} \text{P0()} \quad (\text{Matrix1}, \text{Matrix2}) \Rightarrow \text{matrix}

Returns the equivalent \( \theta \)-coordinate of the \((x,y)\) pair arguments.

\textbf{Note:} The result is returned as a degree, gradians or radian angle, according to the current angle mode setting.

\text{In Degree angle mode:}
\begin{align*}
\text{R} \text{P0}(x,y) & \quad \text{ENTER} \\
\text{R} \text{P0}(3,4) & \quad \text{ENTER} \\
\text{R} \text{P0}((3,-4,2),(0,\pi/4,1.5)) & \quad \text{ENTER}
\end{align*}

\text{In Gradian angle mode:}
\begin{align*}
\text{R} \text{P0}(x,y) & \quad \text{ENTER} \\
\text{R} \text{P0}(3,4) & \quad \text{ENTER} \\
\text{R} \text{P0}((3,-4,2),(0,\pi/4,1.5)) & \quad \text{ENTER}
\end{align*}

\text{In Radian angle mode:}
\begin{align*}
\text{R} \text{P0}(3,4) & \quad \text{ENTER} \\
\text{R} \text{P0}(x,y) & \quad \text{ENTER} \\
\text{R} \text{P0}((3,-4,2),(0,\pi/4,1.5)) & \quad \text{ENTER}
\end{align*}

\text{R} \text{PPr()} \quad \text{MATH/Angle menu}

\text{R} \text{PPr()} \quad (\text{xExpression}, \text{yExpression}) \Rightarrow \text{expression}
\text{R} \text{PPr()} \quad (\text{List1}, \text{List2}) \Rightarrow \text{list}
\text{R} \text{PPr()} \quad (\text{Matrix1}, \text{Matrix2}) \Rightarrow \text{matrix}

Returns the equivalent \( r \)-coordinate of the \((x,y)\) pair arguments.

\text{In Radian angle mode:}
\begin{align*}
\text{R} \text{PPr}(3,4) & \quad \text{ENTER} \\
\text{R} \text{PPr}(x,y) & \quad \text{ENTER} \\
\text{R} \text{PPr}((3,-4,2),(0,\pi/4,1.5)) & \quad \text{ENTER}
\end{align*}

\text{R} \text{Pr()} \quad \text{MATH/Angle menu}

\text{R} \text{Pr()} \quad (\text{xExpression}, \text{yExpression}) \Rightarrow \text{expression}
\text{R} \text{Pr()} \quad (\text{List1}, \text{List2}) \Rightarrow \text{list}
\text{R} \text{Pr()} \quad (\text{Matrix1}, \text{Matrix2}) \Rightarrow \text{matrix}

Returns the equivalent \( \theta \)-coordinate of the \((x,y)\) pair arguments.

\text{In Degree angle mode:}
\begin{align*}
\text{R} \text{Pr}(x,y) & \quad \text{ENTER} \\
\text{R} \text{Pr}(3,4) & \quad \text{ENTER} \\
\text{R} \text{Pr}((3,-4,2),(0,\pi/4,1.5)) & \quad \text{ENTER}
\end{align*}

\text{In Gradian angle mode:}
\begin{align*}
\text{R} \text{Pr}(x,y) & \quad \text{ENTER} \\
\text{R} \text{Pr}(3,4) & \quad \text{ENTER} \\
\text{R} \text{Pr}((3,-4,2),(0,\pi/4,1.5)) & \quad \text{ENTER}
\end{align*}

\text{In Radian angle mode:}
\begin{align*}
\text{R} \text{Pr}(x,y) & \quad \text{ENTER} \\
\text{R} \text{Pr}(3,4) & \quad \text{ENTER} \\
\text{R} \text{Pr}((3,-4,2),(0,\pi/4,1.5)) & \quad \text{ENTER}
\end{align*}

\text{\textbf{\textup{\textnumero Rad}} \quad \text{CATALOG/MATH/Angle menu}}

\text{\textbf{\textnumero Rad} expression}

Converts an expression to radian angle measure.

\text{In Degree angle mode:}
\begin{align*}
1.5 \quad \text{\textnumero Rad} & \quad \text{ENTER} \\
0.02618^\circ & \quad \text{ENTER}
\end{align*}

\text{In Gradian angle mode:}
\begin{align*}
1.5 \quad \text{\textnumero Rad} & \quad \text{ENTER} \\
0.023562^\circ & \quad \text{ENTER}
\end{align*}

\text{\textbf{\textup{\textnumero Rand}} \quad \text{MATH/Probability menu}}

\text{\textnumero Rand()} \Rightarrow \text{expression}

\begin{align*}
\text{RandSeed} & \quad 1147 \quad \text{ENTER} \\
\text{Done} & \quad \text{SET} \quad \text{(Sets the random-number)}
\end{align*}

With no parameter, returns the next random number between 0 and 1 in the sequence. When an argument is positive, returns a random integer in the interval \([1, n]\). When an argument is negative, returns a random integer in the interval \([- n, -1]\).

\begin{align*}
\text{rand()} & \quad \text{ENTER} \\
.158... & \quad \text{ENTER} \\
\text{rand}(6) & \quad \text{ENTER} \\
5 & \quad \text{ENTER} \\
\text{rand}(-100) & \quad \text{ENTER} \\
-49 & \quad \text{ENTER}
\end{align*}
**randMat()** MATH/Probability menu

**randMat**(numRows, numColumns) ⇒ matrix

Returns a matrix of integers between -9 and 9 of the specified dimension.

Both arguments must simplify to integers.

Note: The values in this matrix will change each time you press [ENTER].

**randNorm()** MATH/Probability menu

**randNorm**(mean, sd) ⇒ expression

Returns a decimal number from the specific normal distribution. It could be any real number but will be heavily concentrated in the interval

\[ [\text{mean}-3 \times \text{sd}, \text{mean}+3 \times \text{sd}] \]

**randPoly()** MATH/Probability menu

**randPoly**(var, order) ⇒ expression

Returns a polynomial in var of the specified order. The coefficients are random integers in the range -9 through 9. The leading coefficient will not be zero.

order must be 0–99.

**RandSeed** MATH/Probability menu

**RandSeed** number

If number = 0, sets the seeds to the factory defaults for the random-number generator. If number ≠ 0, it is used to generate two seeds, which are stored in system variables seed1 and seed2.

**RclGDB** CATALOG

**RclGDB** GDBvar

Restores all the settings stored in the Graph database variable GDBvar.

For a listing of the settings, see **StoGDB**.

Note: It is necessary to have something saved in GDBvar before you can restore it.

**RclPic** CATALOG

**RclPic** picVar[row, column]

Displays the Graph screen and adds the picture stored in picVar at the upper left-hand corner pixel coordinates (row, column) using OR logic.

picVar must be a picture data type.

Default coordinates are (0, 0).

**real()** MATH/Complex menu

**real**(expression1) ⇒ expression

Returns the real part of the argument.

Note: All undefined variables are treated as real variables. See also **imag()**.
real(list) ⇒ list

Returns the real parts of all elements.

real(matrix) ⇒ matrix

Returns the real parts of all elements.

Rect

Displays \( \text{vector} \) in rectangular form \([x, y, z] \). The vector must be of dimension 2 or 3 and can be a row or a column.

Note: Rect is a display-format instruction, not a conversion function. You can use it only at the end of an entry line, and it does not update \( \text{ans} \).

Note: See also Polar.

complexValue Rect

Displays complexValue in rectangular form \( a + bi \).

The complexValue can have any complex form. However, an \( r \theta \) entry causes an error in Degree angle mode.

Note: You must use parentheses for an \( r \theta \) polar entry.

In Radian angle mode:

\[
4e^{i(\pi/3)} \text{Rect} \quad 4 \cdot \text{e}^{i\pi/3}
\]

\[
(4\pi/3) \text{Rect} \quad 2 + 2\sqrt{3}i
\]

In Gradian angle mode:

\[
(1\,100) \text{Rect} \quad i
\]

In Degree angle mode:

\[
(4\,60) \text{Rect} \quad 2 + 2\sqrt{3}i
\]

Note: To type Rect from the keyboard, press \( 2 \Rightarrow \) for the \( \text{operator}\). To type \( \theta \), press \( 2'\).

ref()

Returns the row echelon form of matrix1. Optionally, any matrix element is treated as zero if its absolute value is less than tol. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, tol is ignored.

- If you use \( \text{Approx} \) or set the mode to Exact/Approx=APPROXIMATE, computations are done using floating-point arithmetic.
- If \( \text{tol} \) is omitted or not used, the default tolerance is calculated as:

\[
5x \times 14 + \max(\text{dim(matrix)}) \times \text{rowNorm(matrix)}
\]

Note: See also \( \text{ref()} \).
### remain() MATH/Number menu

\[
\text{remain}(\text{expression}_1, \text{expression}_2) \Rightarrow \text{expression}_1 - \text{expression}_2
\]

\[
\text{remain}(\text{list}_1, \text{list}_2) \Rightarrow \text{list}_1 - \text{list}_2
\]

\[
\text{remain}(\text{matrix}_1, \text{matrix}_2) \Rightarrow \text{matrix}_1 - \text{matrix}_2
\]

Returns the remainder of the first argument with respect to the second argument as defined by the identities:

\[
\text{remain}(x, 0) = x
\]

\[
\text{remain}(x, y) = x - y \cdot \text{iPart}(x/y)
\]

As a consequence, note that \(\text{remain}(-x, y) = -\text{remain}(x, y)\). The result is either zero or it has the same sign as the first argument.

**Note:** See also `mod()`.

### Rename CATALOG

**Rename** oldVarName, newVarName

Renames the variable oldVarName as newVarName.

### Request CATALOG

**Request** promptString, var [ , alphaOn/Off ]

If `Request` is inside a Dialog...EndDlog construct, it creates an input box for the user to type in data. If it is a stand-alone instruction, it creates a dialog box for this input. In either case, if `var` contains a string, it is displayed and highlighted in the input box as a default choice. `promptString` must be ≤ 20 characters.

This instruction can be stand-alone or part of a dialog construct.

The optional `alphaOn/Off` argument can be any expression. If it evaluates to zero, alpha-lock is set to OFF. If it evaluates to anything other than zero, alpha-lock is set to ON. If the optional argument is not used, alpha-lock defaults to ON.

If more than one `Request` command appears within a Dialog...EndDlog construct, the first alpha setting is used and subsequent ones are ignored.

### Return CATALOG

**Return** [expression]

Returns `expression` as the result of the function. Use within a `Func...EndFunc` block, or `Prgm...EndPrgm` block.

**Note:** Use `Return` without an argument to exit a program.

**Note:** Enter the text as one long line on the Home screen (without line breaks).
right()  MATH/List menu
right(list1, num) ⇒ list

Returns the rightmost num elements contained in list1.
If you omit num, returns all of list1.

right(sourceString, num) ⇒ string

Returns the rightmost num characters contained in character string sourceString.
If you omit num, returns all of sourceString.

right(comparison) ⇒ expression

Returns the right side of an equation or inequality.

root()  CATALOG/MATH/Number menu
root(expression) ⇒ root

Computes an nth root of x where x can be a real or complex floating point constant, an integer or complex rational constant, or a general symbolic expression.

rotate()  MATH/Base menu
rotate(integer1, #ofRotations) ⇒ integer

Rotates the bits in a binary integer. You can enter integer1 in any number base; it is converted automatically to a signed, 32-bit binary form. If the magnitude of integer1 is too large for this form, a symmetric modulo operation brings it within the range.

If #ofRotations is positive, the rotation is to the left. If #ofRotations is negative, the rotation is to the right. The default is -1 (rotate right one bit).

For example, in a right rotation:

In Bin base mode:

rotate(0b11110110110000110101) ⇒ 0b10000000000000111101011000011010

rotate(256,1) ⇒ 0b100000000

rotate(0h78E) ⇒ 0h3C7

rotate(0h78E, -2) ⇒ 0h800001E3

rotate(0h78E, 2) ⇒ 0h1E38

Important: To enter a binary or hexadecimal number, always use the 0b or 0h prefix (zero, not the letter O).

rotate(list1, #ofRotations) ⇒ list

Returns a copy of list1 rotated right or left by #ofRotations elements. Does not alter list1.

If #ofRotations is positive, the rotation is to the left. If #ofRotations is negative, the rotation is to the right. The default is -1 (rotate right one element).
rotate(string1,#ofRotations) ⇒ string

Returns a copy of string1 rotated right or left by #ofRotations characters. Does not alter string1.
If #ofRotations is positive, the rotation is to the left. If #ofRotations is negative, the rotation is to the right. The default is -1 (rotate right one character).

\[
\begin{align*}
\text{rotate("abcd") } & \text{ ENTER} \quad \text{"dabc"} \\
\text{rotate("abcd",-2) } & \text{ ENTER} \quad \text{"cdab"} \\
\text{rotate("abcd",1) } & \text{ ENTER} \quad \text{"bcda"}
\end{align*}
\]

round() MATH/Number menu

round(expression1[digits]) ⇒ expression

Returns the argument rounded to the specified number of digits after the decimal point.
digits must be an integer in the range 0–12. If digits is not included, returns the argument rounded to 12 significant digits.

Note: Display digits mode may affect how this is displayed.

\[
\begin{align*}
\text{round(1.234567,3) } & \text{ ENTER} \quad 1.235 \\
\text{round([}\pi,\sqrt{2},\ln(2)\],4) } & \text{ ENTER} \quad (3.1416 \ 1.4142 \ 0.6931)
\end{align*}
\]

rowAdd() MATH/Matrix/Row ops menu

rowAdd(matrix1, rIndex1, rIndex2) ⇒ matrix

Returns a copy of matrix1 with row rIndex2 replaced by the sum of rows rIndex1 and rIndex2.

\[
\begin{align*}
\text{rowAdd([}3,4;\ 3,2\],1,2) } & \text{ ENTER} \quad \begin{bmatrix} 3 & 4 \\ 0 & 2 \end{bmatrix} \\
\text{rowAdd([}a,b;c,d\],1,2) } & \text{ ENTER} \quad \begin{bmatrix} a+c \ a \\ b \ b+d \end{bmatrix}
\end{align*}
\]

rowDim() MATH/Matrix/Dimensions menu

rowDim(matrix) ⇒ expression

Returns the number of rows in matrix.

Note: See also colDim().

\[
\begin{align*}
\text{rowDim([}1,2;3,4;5,6\]) } & \text{ ENTER} \quad 3 \\
\text{rowDim(M1) } & \text{ ENTER} \quad 3
\end{align*}
\]

rowNorm() MATH/Matrix/Norms menu

rowNorm(matrix) ⇒ expression

Returns the maximum of the sums of the absolute values of the elements in the rows in matrix.

Note: All matrix elements must simplify to numbers. See also colNorm().

\[
\begin{align*}
\text{rowNorm([}5.6..7;3.4;9.9..7\]) } & \text{ ENTER} \quad 25
\end{align*}
\]
rowSwap()  MATH/Matrix/Row ops menu

rowSwap(matrix1, rIndex1, rIndex2) ⇒ matrix

Returns matrix1 with rows rIndex1 and rIndex2 exchanged.

rowSwap(Mat, 1, 3)  ENTER

| 5 6 |
|---|---|
| 3 4 |
| 1 2 |

RplcPic  CATALOG

RplcPic picVar, row[, column]

Clears the Graph screen and places picture picVar at pixel coordinates (row, column). If you do not want to clear the screen, use RclPic.

picVar must be a picture data type variable. row and column, if included, specify the pixel coordinates of the upper left corner of the picture.

Default coordinates are (0, 0).

Note: For less than full-screen pictures, only the area affected by the new picture is cleared.

rref()  MATH/Matrix menu

rref(matrix1[, tol]) ⇒ matrix

Returns the reduced row echelon form of matrix1.

rref([[1,2;3,4;5,6]] ENTER

| 1 0 |
|---|---|
| 0 1 |
| 0 0 |

Optionally, any matrix element is treated as zero if its absolute value is less than tol. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, tol is ignored.

• If you use ENTER or set the mode to Exact/Approx=APPROXIMATE, computations are done using floating-point arithmetic.

• If tol is omitted or not used, the default tolerance is calculated as:

\[
5 \times 10^{-14} \times \max(\text{dim}(\text{matrix1})) \times \text{rowNorm}(\text{matrix1})
\]

Note: See also ref().

sec()  MATH/Trig menu

sec(expression) ⇒ expression
sec(list1) ⇒ list

Returns the secant of expression1 or returns a list containing the secants of all elements in list1.

Note: The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.
sec⁻¹() \hspace{1cm} \text{MATH/Trig menu}

\begin{align*}
\text{sec}^{-1}(\text{expression}) & \Rightarrow \text{expression} \\
\text{sec}^{-1}(\text{list}) & \Rightarrow \text{list}
\end{align*}

Returns the angle whose secant is expression1 or returns a list containing the inverse secants of each element of list1.

\textbf{Note:} The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

In Degree angle mode:

\[
\text{sec}^{-1}(1) \ \text{ENTER} \quad 0
\]

In Gradian angle mode:

\[
\text{sec}^{-1}(\sqrt{2}) \ \text{ENTER} \quad 50
\]

In Radian angle mode:

\[
\text{sec}^{-1}([1,2,5]) \ \text{ENTER} \quad \{0 \ \cos^{-1}(1/5)\}
\]

sech() \hspace{1cm} \text{MATH/Hyperbolic menu}

\begin{align*}
\text{sech}(\text{expression}) & \Rightarrow \text{expression} \\
\text{sech}(\text{list}) & \Rightarrow \text{list}
\end{align*}

Returns the hyperbolic secant of expression1 or returns a list containing the hyperbolic secants of the list1 elements.

\[
\text{sech}(3) \ \text{ENTER} \quad \frac{1}{\cosh(3)}
\]

\[
\text{sech}([1,2,3,4]) \ \text{ENTER} \quad \left(\frac{1}{\cosh(1)} \cdot 108 \cdot \frac{1}{\cosh(4)}\right)
\]

sech⁻¹() \hspace{1cm} \text{MATH/Hyperbolic menu}

\begin{align*}
\text{sech}^{-1}(\text{expression}) & \Rightarrow \text{expression} \\
\text{sech}^{-1}(\text{list}) & \Rightarrow \text{list}
\end{align*}

Returns the inverse hyperbolic secant of expression1 or returns a list containing the inverse hyperbolic secants of each element of list1.

In Radian angle and Rectangular complex mode:

\[
\text{sech}^{-1}(1) \ \text{ENTER} \quad 0
\]

\[
\text{sech}^{-1}([1,2,2,1]) \ \text{ENTER} \quad \{0 \ \frac{2\pi}{3} \ \cdot \ i \ 1.074\ldots \cdot i\}
\]

\textbf{Send} \hspace{1cm} \text{CATALOG}

\textbf{Send} \hspace{1cm} \text{list}

CBL 2™ (Calculator-Based Laboratory™) or CBR™ (Calculator-Based Ranger™) instruction. Sends list to the link port.

Program segment:

\[
: \text{Send} \ {\{1.0\}} \\
: \text{Send} \ {\{1.2.1\}} \\
:
\]

\textbf{SendCalc} \hspace{1cm} \text{CATALOG}

\textbf{SendCalc} \hspace{1cm} \text{var}

Sends variable var to the link port, where another unit linked to that port can receive the variable value. The receiving unit must be on the Home screen or must execute \texttt{GetCalc} from a program.

If you send from a TI-89, TI-92 Plus, or Voyage™ 200 to a TI-92, an error occurs if the TI-92 executes \texttt{GetCalc} from a program. In this case, the sending unit must use \texttt{SendChat} instead.
**SendCalc var[,port]**

Sends contents of `var` from a TI-89 Titanium to another TI-89 Titanium.

If the port is not specified, or `port = 0` is specified, the TI-89 Titanium sends data using the USB port if connected, if not, it will send using the I/O port.

If `port = 1`, the TI-89 Titanium sends data using the USB port only.

If `port = 2`, the TI-89 Titanium sends data using the I/O port only.

**SendChat CATALOG**

**SendChat var**

A general alternative to `SendCalc`, this is useful if the receiving unit is a TI-92 (or for a generic "chat" program that allows either a TI-92, Voyage™ 200, or TI-92 Plus to be used). Refer to `SendCalc` for more information.

`SendChat` sends a variable only if that variable is compatible with the TI-92, which is typically true in "chat" programs. However, `SendChat` will not send an archived variable, a TI-89 graph data base, etc.

**seq() MATH/List menu**

`seq(expression, var, low, high[, step])` ⇒ list

Increments `var` from `low` through `high` by an increment of `step`, evaluates `expression`, and returns the results as a list. The original contents of `var` are still there after `seq()` is completed.

`var` cannot be a system variable.

The default value for `step = 1`.

**setDate() CATALOG**

`setDate(year,month,day)` ⇒ listold

Sets the clock to the date given in the argument and returns a list. (Note: The year must fall in the range 1997 - 2132.) The returned list is in `{yearold,monthold,dayold}` format. The returned date is the previous clock value.

Enter the year as a four-digit integer. The month and day can be either one- or two-digit integers.

**setDtFmt() CATALOG**

`setDtFmt(integer)` ⇒ integerold

Sets the date format for the desktop according to the argument and returns the previous date format value.

**Program segment:**

```
:a+b\x
:SendChat \x
```

**Example:**

```
seq(n^2,n,1,6) ENTER
{1 4 9 16 25 36}

seq(1/n,n,1,10,2) ENTER
{1 1/3 1/5 1/7 1/9}

sum(seq(1\n^2,n,1,10,1)) ENTER
196...
127...
```

or press \[ENTER] to get: 1.549...

**Example:**

```
setDate(2001,10,31)
{2001 11 1}
```

**Example:**

```
setDtFmt(1) ENTER
```

**Integer values:**

1 = MM/DD/YY  5 = YY.MM.DD
2 = DD/MM/YY  6 = MM-DD-YY
3 = MM.DD.YY  7 = DD-MM-YY
4 = DD.MM.YY  8 = YY-MM-DD

Appendix A: Functions and Instructions 869
### setFold()  CATALOG

```plaintext
setFold(newfolderName) ⇒ oldfolderString
```

Returns the name of the current folder as a string and sets newfolderName as the current folder.

The folder newfolderName must exist.

```plaintext
setFold(chris) ⇒ oldfoldr
```

```
> a ENTER
1
```

```plaintext
setFold(#oldfoldr) ⇒ chris
```

```
a ENTER
```

The folder newfolderName must exist.

### setGraph()  CATALOG

```plaintext
setGraph(modeNameString, settingString) ⇒ string
```

Sets the Graph mode modeNameString to settingString, and returns the previous setting of the mode. Storing the previous setting lets you restore it later.

- **modeNameString** is a character string that specifies which mode you want to set. It must be one of the mode names from the table below.
- **settingString** is a character string that specifies the new setting for the mode. It must be one of the settings listed below for the specific mode you are setting.

Note: Capitalization and blank spaces are optional when entering mode names.

<table>
<thead>
<tr>
<th>Mode Name</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Coordinates</em></td>
<td>“Rect”, “Polar”, “Off”</td>
</tr>
<tr>
<td><em>Graph Order</em></td>
<td>“Seq”, “Simul” ¹</td>
</tr>
<tr>
<td><em>Grid</em></td>
<td>“Off”, “On” ²</td>
</tr>
<tr>
<td><em>Axes</em></td>
<td>“Off”, “On” (not 3D graph mode)</td>
</tr>
<tr>
<td></td>
<td>“Off”, “Axes”, “Box” (3D graph mode)</td>
</tr>
<tr>
<td><em>Leading Cursor</em></td>
<td>“Off”, “On” ²</td>
</tr>
<tr>
<td><em>Labels</em></td>
<td>“Off”, “On”</td>
</tr>
<tr>
<td><em>Style</em></td>
<td>“Wire Frame”, “Hidden Surface”, “Contour Levels”, “Wire and Contour”, “Implicit Plot” ³</td>
</tr>
<tr>
<td><em>Seq Axes</em></td>
<td>“Time”, “Web”, “U1-vs-U2” ⁴</td>
</tr>
<tr>
<td><em>DE Axes</em></td>
<td>“Time”, “t-vs-y’”, “y-vs-y’”, “y1-vs-y2”, “y1-vs-y2’”, “y1’-vs-y2’” ⁵</td>
</tr>
<tr>
<td><em>Solution Method</em></td>
<td>“RK”, “Euler” ⁵</td>
</tr>
<tr>
<td><em>Fields</em></td>
<td>“SlpFld”, “DirFld”, “FldOff” ⁵</td>
</tr>
<tr>
<td><em>Discontinuity Detection</em></td>
<td>“Off”, “On” ⁶</td>
</tr>
</tbody>
</table>

¹Not available in Sequence, 3D, or Diff Equations graph mode. Also not available in Function graph mode with “Discontinuity Detection” set to “On.”
²Not available in 3D graph mode.
³To type a prime symbol (’), press 2nd [‘].
setMode() CATALOG

`setMode(modeNameString, settingString) ⇒ string`
`setMode(list) ⇒ stringlist`

Sets mode `modeNameString` to the new setting `settingString`, and returns the current setting of that mode.

`modeNameString` is a character string that specifies which mode you want to set. It must be one of the mode names from the table below.
`settingString` is a character string that specifies the new setting for the mode. It must be one of the settings listed below for the specific mode you are setting.
`list` contains pairs of keyword strings and will set them all at once. This is recommended for multiple-mode changes. The example shown may not work if each of the pairs is entered with a separate `setMode()` in the order shown.

Use `setMode(var)` to restore settings saved with `getMode("ALL") ⇒ var`.

**Note:** To set or return information about the Unit System mode, use `setUnits()` or `getUnits()` instead of `setMode()` or `getMode()`.

<table>
<thead>
<tr>
<th>Mode Name</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Graph&quot;</td>
<td>&quot;Function&quot;, &quot;Parametric&quot;, &quot;Polar&quot;, &quot;Sequence&quot;, &quot;3D&quot;, &quot;Diff Equations&quot;</td>
</tr>
<tr>
<td>&quot;Display Digits&quot;</td>
<td>&quot;Fix 0&quot;, &quot;Fix 1&quot;, ..., &quot;Fix 12&quot;, &quot;Float&quot;, &quot;Float 1&quot;, ..., &quot;Float 12&quot;</td>
</tr>
<tr>
<td>&quot;Angle&quot;</td>
<td>&quot;Radian&quot;, &quot;Degree&quot;, &quot;Gradian&quot;</td>
</tr>
<tr>
<td>&quot;Exponential Format&quot;</td>
<td>&quot;Normal&quot;, &quot;Scientific&quot;, &quot;Engineering&quot;</td>
</tr>
<tr>
<td>&quot;Complex Format&quot;</td>
<td>&quot;Real&quot;, &quot;Rectangular&quot;, &quot;Polar&quot;</td>
</tr>
<tr>
<td>&quot;Vector Format&quot;</td>
<td>&quot;Rectangular&quot;, &quot;Cylindrical&quot;, &quot;Spherical&quot;</td>
</tr>
<tr>
<td>&quot;Pretty Print&quot;</td>
<td>&quot;Off&quot;, &quot;On&quot;</td>
</tr>
</tbody>
</table>

Appendix A: Functions and Instructions 871
### Functions and Instructions

#### "Split Screen"
- "Full", "Top-Bottom", "Left-Right"

#### "Split 1 App"
- "Home", "Y= Editor", "Window Editor", "Graph", "Table", "Data/Matrix Editor", "Program Editor", "Text Editor", "Numeric Solver", "Flash App"

#### "Split 2 App"
- "Home", "Y= Editor", "Window Editor", "Graph", "Table", "Data/Matrix Editor", "Program Editor", "Text Editor", "Numeric Solver", "Flash App"

#### "Number of Graphs"
- "1", "2"

#### "GraphType"

#### "Split Screen Ratio"
- "1:1", "1:2", "2:1"  (Voyage™ 200 only)

#### "Exact/Approx"
- "Auto", "Exact", "Approximate"

#### "Base"
- "Dec", "Hex", "Bin"

#### "Language"
- "English", "Alternate Language"

#### "Apps Desktop"
- "Off", "On"

---

### setTable()

**Syntax**

```plaintext
setTable(modeNameString, settingString) ⇒ string
```

**Description**

Sets the table parameter `modeNameString` to `settingString`, and returns the previous setting of the parameter. Storing the previous setting lets you restore it later.

- **modeNameString** is a character string that specifies which parameter you want to set. It must be one of the parameters from the table below.
- **settingString** is a character string that specifies the new setting for the parameter. It must be one of the settings listed below for the specific parameter you are setting.

**Parameter Name**
- "Graph <-> Table"
- "Independent"

**Settings**
- "Off", "On"
- "Auto", "Ask"

---

### setTime()

**Syntax**

```plaintext
setTime(hour, minute, second) ⇒ list
```

**Description**

Sets the clock to the time given in the argument and returns a list. The list is in `{hourold, minuteold, secondold}` format. The returned time is the previous clock value.

**Example**

```plaintext
setTime(11, 32, 50)  ⇒ {10, 44, 49}
```

**Note:** Capitalization and blank spaces are optional when entering parameters.

---

### setTmFmt()

**Syntax**

```plaintext
setTmFmt(integer) ⇒ integerold
```

**Description**

Sets the time format for the desktop according to the argument and returns the previous time format value.

**Integer values:**
- 12 = 12 hour clock
- 24 = 24 hour clock
**setTmZn() CATALOG**

```plaintext
setTmZn(integer) ⇒ integerold
```

Sets the time zone according to the argument and returns the previous time zone value.

The time zone is defined by an integer that gives the minutes offset from Greenwich Mean Time (GMT), as established in Greenwich, England. For example, if the time zone is offset from GMT by two hours, the device would return 120 (minutes).

Integers for time zones west of GMT are negative.
Integers for time zones east of GMT are positive.

If Greenwich Mean Time is 14:07:07, it is:

- 7:07:07 a.m. in Denver, Colorado (Mountain Standard Time) 
  (−420 minutes from GMT)
- 15:07:07 p.m. in Brussels, Belgium (Central European Standard Time) 
  (+60 minutes from GMT)

**setUnits() CATALOG**

```plaintext
setUnits(list) ⇒ list
```

Sets the default units to the values specified in list, and returns a list of the previous defaults.

- To specify the built-in SI (metric) or ENG/US system, list uses the form:
  ```plaintext
  {"SI"}   or   {"ENG/US"}
  ```
- To specify a custom set of default units, list uses the form:
  ```plaintext
  {"CUSTOM", "cat1", "unit1"[ , "cat2", "unit2", ...]}
  ```

  where each cat and unit pair specifies a category and its default unit. (You can specify built-in units only, not user-defined units.) Any category not specified will use its previous custom unit.

- To return to the previous custom default units, list uses the form:
  ```plaintext
  {"CUSTOM"}
  ```

If you want different defaults depending on the situation, create separate lists and save them to unique list names. To use a set of defaults, specify that list name in `setUnits()`.

You can use `setUnits()` to restore settings previously saved with `setUnits()` + `var` or with `getUnits()` + `var`.

All unit names must begin with an underscore _.

You can also select units from a menu by pressing:

<table>
<thead>
<tr>
<th>Menu Items</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>29 Units</td>
<td>Enter</td>
</tr>
<tr>
<td>setUnits({&quot;SI&quot;})</td>
<td>Enter</td>
</tr>
<tr>
<td>setUnits({&quot;CUSTOM&quot;,&quot;Length&quot;,&quot;_cm&quot;,&quot;Mass&quot;,&quot;_gm&quot;})</td>
<td>Enter</td>
</tr>
</tbody>
</table>

Note: Your screen may display different units.
Shade \texttt{CATALOG}

\textbf{Shade} \textit{expr1, expr2, [xlow], [xhigh], [pattern], [patRes]}

Displays the Graph screen, graphs \textit{expr1} and \textit{expr2}, and shades areas in which \textit{expr1} is less than \textit{expr2}. (\textit{expr1} and \textit{expr2} must be expressions that use \textit{x} as the independent variable.)

\textit{xlow} and \textit{xhigh}, if included, specify left and right boundaries for the shading. Valid inputs are between \textit{xmin} and \textit{xmax}. Defaults are \textit{xmin} and \textit{xmax}.

\textit{pattern} specifies one of four shading patterns:
\begin{itemize}
  \item 1 = vertical (default)
  \item 2 = horizontal
  \item 3 = negative-slope 45°
  \item 4 = positive-slope 45°
\end{itemize}

\textit{patRes} specifies the resolution of the shading patterns:
\begin{itemize}
  \item 1 = solid shading
  \item 2 = 1 pixel spacing (default)
  \item 3 = 2 pixels spacing
  \item 10 = 9 pixels spacing
\end{itemize}

\textbf{Note:} Interactive shading is available on the Graph screen through the \textbf{Shade} instruction. Automatic shading of a specific function is available through the \textbf{Style} instruction. \textbf{Shade} is not valid in 3D graphing mode.

\begin{itemize}
  \item In the ZoomTrig viewing window:
    \begin{itemize}
      \item \texttt{Shade cos(x),sin(x) \textbf{[ENTER]}}
        \begin{itemize}
          \item \texttt{ClrDraw \textbf{[ENTER]}}
            \item \texttt{Done}
        \end{itemize}
      \item \texttt{Shade cos(x),sin(x),0,5 \textbf{[ENTER]}}
        \begin{itemize}
          \item \texttt{ClrDraw \textbf{[ENTER]}}
            \item \texttt{Done}
        \end{itemize}
      \item \texttt{Shade cos(x),sin(x),0,5,2 \textbf{[ENTER]}}
        \begin{itemize}
          \item \texttt{ClrDraw \textbf{[ENTER]}}
            \item \texttt{Done}
        \end{itemize}
      \item \texttt{Shade cos(x),sin(x),0,5,2,1 \textbf{[ENTER]}}
        \begin{itemize}
          \item \texttt{ClrDraw \textbf{[ENTER]}}
            \item \texttt{Done}
        \end{itemize}
    \end{itemize}
\end{itemize}

shift() \texttt{CATALOG}

\textbf{shift(\textit{integer1}, \#ofShifts)} \Rightarrow \textit{integer}

Shifts the bits in a binary integer. You can enter \textit{integer1} in any number base; it is converted automatically to a signed, 32-bit binary form. If the magnitude of \textit{integer1} is too large for this form, a symmetric modulo operation brings it within the range.

If \#ofShifts is positive, the shift is to the left. If \#ofShifts is negative, the shift is to the right. The default is “1” (shift right one bit).

In a right shift, the rightmost bit is dropped and 0 or 1 is inserted to match the leftmost bit. In a left shift, the leftmost bit is dropped and 0 is inserted as the rightmost bit.

For example, in a right shift:

\begin{itemize}
  \item In Bin base mode:
    \begin{itemize}
      \item \texttt{shift(0b1111010110000110101) \textbf{[ENTER]}}
        \begin{itemize}
          \item \texttt{0b1111010110000110}
          \item \texttt{0b1000000000}
        \end{itemize}
      \item \texttt{shift(256,1) \textbf{[ENTER]}}
        \begin{itemize}
          \item \texttt{Ob1111010110000110101}
          \item \texttt{Ob1000000000}
        \end{itemize}
    \end{itemize}
  \item In Hex base mode:
    \begin{itemize}
      \item \texttt{shift(0h78E) \textbf{[ENTER]}}
        \begin{itemize}
          \item \texttt{0h3C7}
          \item \texttt{0h1E3}
        \end{itemize}
      \item \texttt{shift(0h78E, -2) \textbf{[ENTER]}}
        \begin{itemize}
          \item \texttt{0h1E3}
          \item \texttt{0h3C7}
        \end{itemize}
    \end{itemize}
\end{itemize}

\textbf{Important:} To enter a binary or hexadecimal number, always use the 0b or 0h prefix (zero, not the letter O).
Each bit shifts right.  
0b0000000000001111010110000110101  
Inserts 0 if leftmost bit is 0,  
or 1 if leftmost bit is 1.  
produces:  
0b000000000000111101011000011010  
The result is displayed according to the Base  
mode. Leading zeros are not shown.

<table>
<thead>
<tr>
<th>shift(list1, #ofShifts) ⇒ list</th>
<th>In Dec base mode:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a copy of list1 shifted right or left by</td>
<td></td>
</tr>
<tr>
<td>#ofShifts elements. Does not alter list1.</td>
<td></td>
</tr>
<tr>
<td>If #ofShifts is positive, the shift is to the left.</td>
<td></td>
</tr>
<tr>
<td>If #ofShifts is negative, the shift is to the right.</td>
<td></td>
</tr>
<tr>
<td>The default is -1 (shift right one element).</td>
<td></td>
</tr>
<tr>
<td>Elements introduced at the beginning or end of</td>
<td></td>
</tr>
<tr>
<td>list by the shift are set to the symbol &quot;undef&quot;.</td>
<td></td>
</tr>
<tr>
<td>shift(string1, #ofShifts) ⇒ string</td>
<td></td>
</tr>
<tr>
<td>Returns a copy of string1 shifted right or left by</td>
<td></td>
</tr>
<tr>
<td>#ofShifts characters. Does not alter string1.</td>
<td></td>
</tr>
<tr>
<td>If #ofShifts is positive, the shift is to the left.</td>
<td></td>
</tr>
<tr>
<td>If #ofShifts is negative, the shift is to the right.</td>
<td></td>
</tr>
<tr>
<td>The default is -1 (shift right one character).</td>
<td></td>
</tr>
<tr>
<td>Characters introduced at the beginning or end of</td>
<td></td>
</tr>
<tr>
<td>string by the shift are set to a space.</td>
<td></td>
</tr>
</tbody>
</table>

**ShowStat**  
Displays a dialog box containing the last  
computed statistics results if they are still valid.  
Statistics results are cleared automatically if the  
data to compute them has changed.  
Use this instruction after a statistics calculation,  
such as LinReg.

<table>
<thead>
<tr>
<th>sign()</th>
<th>MATH/Number menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>sign(expression) ⇒ expression</td>
<td></td>
</tr>
<tr>
<td>sign(list) ⇒ list</td>
<td></td>
</tr>
<tr>
<td>sign(matrix) ⇒ matrix</td>
<td></td>
</tr>
<tr>
<td>For real and complex expression, returns</td>
<td></td>
</tr>
<tr>
<td>expression/abs(expression) when expression≠ 0.</td>
<td></td>
</tr>
<tr>
<td>Returns 1 if expression is positive.</td>
<td></td>
</tr>
<tr>
<td>Returns -1 if expression is negative.</td>
<td></td>
</tr>
<tr>
<td>sign(0) returns ±1 if the complex format mode is</td>
<td></td>
</tr>
<tr>
<td>REAL; otherwise, it returns itself.</td>
<td></td>
</tr>
<tr>
<td>sign(0) represents the unit circle in the complex</td>
<td></td>
</tr>
<tr>
<td>domain.</td>
<td></td>
</tr>
<tr>
<td>For a list or matrix, returns the signs of all the</td>
<td></td>
</tr>
<tr>
<td>elements.</td>
<td></td>
</tr>
</tbody>
</table>

**Appendix A: Functions and Instructions**  
875
**simult()**

**MATH/Matrix menu**

**simult(coeffMatrix, constVector, tol)** ⇒ matrix

Returns a column vector that contains the solutions to a system of linear equations.

- `coeffMatrix` must be a square matrix that contains the coefficients of the equations.
- `constVector` must have the same number of rows (same dimension) as `coeffMatrix` and contain the constants.

Optionally, any matrix element is treated as zero if its absolute value is less than `tol`. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, `tol` is ignored.

- If you use `ckeck` or set the mode to Exact/Approx=APPROXIMATE, computations are done using floating-point arithmetic.
- If `tol` is omitted or not used, the default tolerance is calculated as:

\[
5 \times 10^{-14} + \max(\text{dim}(\text{coeffMatrix})) \times \text{rowNorm}(\text{coeffMatrix})
\]

**simult(coeffMatrix, constMatrix, tol)** ⇒ matrix

Solves multiple systems of linear equations, where each system has the same equation coefficients but different constants.

Each column in `constMatrix` must contain the constants for a system of equations. Each column in the resulting matrix contains the solution for the corresponding system.

- `coeffMatrix` must be a square matrix that contains the coefficients of the equations.
- `constMatrix` must have the same number of rows (same dimension) as `coeffMatrix` and contain the constants.

Optionally, any matrix element is treated as zero if its absolute value is less than `tol`. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, `tol` is ignored.

- If you use `ckeck` or set the mode to Exact/Approx=APPROXIMATE, computations are done using floating-point arithmetic.
- If `tol` is omitted or not used, the default tolerance is calculated as:

\[
5 \times 10^{-14} + \max(\text{dim}(\text{coeffMatrix})) \times \text{rowNorm}(\text{coeffMatrix})
\]

---

**sin()**

**\(2W\) key**

**sin(expression)** ⇒ expression

**sin(list)** ⇒ list

- **sin(expression)** returns the sine of the argument as an expression.
- **sin(list)** returns a list of the sines of all elements in `list`.

**Note:** The argument is interpreted as a degree, gradian or radian angle, according to the current angle mode. You can use `°`, `\(^\circ\)` or `\(^\circ\)` to override the angle mode setting temporarily.

In Degree mode:

- \(\sin(\pi/4)\) ENTER \[\sqrt{2} \over 2\]
- \(\sin(45)\) ENTER \[\sqrt{2} \over 2\]

In Gradian mode:

- \(\sin(50)\) ENTER \[\sqrt{2} \over 2\]

In Radian mode:

- \(\sin(\pi/4)\) ENTER \[\sqrt{2} \over 2\]
- \(\sin(45^\circ)\) ENTER \[\sqrt{2} \over 2\]}
\[
\sin(\text{squareMatrix1}) \Rightarrow \text{squareMatrix}
\]

Returns the matrix sine of \text{squareMatrix1}. This is not the same as calculating the sine of each element. For information about the calculation method, refer to \(\cos()\).

\text{squareMatrix1} must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode:
\[
\sin([1,5;3,4;2,1;6,-2,1]) \quad \text{ENTER}
\]
\[
\begin{bmatrix}
.942... & -.045... & -.031...
\end{bmatrix}
\]
\[
\begin{bmatrix}
.045... & .949... & .020...
\end{bmatrix}
\]
\[
\begin{bmatrix}
.048... & -.005... & .961...
\end{bmatrix}
\]

\(\sin^{-1}()\)

\[
\sin^{-1}(\text{expression}) \Rightarrow \text{expression}
\]

\[
\sin^{-1}(\text{list}) \Rightarrow \text{list}
\]

\(\sin^{-1}(\text{expression})\) returns the angle whose sine is \text{expression} as an expression.

\(\sin^{-1}(\text{list})\) returns a list of the inverse sines of each element of \text{list}.

**Note:** The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

In Degree angle mode:
\[
\sin^{-1}(1) \quad \text{ENTER}
\]
\[
90
\]

In Gradian angle mode:
\[
\sin^{-1}(1) \quad \text{ENTER}
\]
\[
100
\]

In Radian angle mode:
\[
\sin^{-1}([0.2, 0.5]) \quad \text{ENTER}
\]
\[
\begin{bmatrix}
0.201... & .523... \end{bmatrix}
\]

\(\sinh()\)

\[\text{MATH/Hyperbolic menu}\]

\[
\sinh(\text{expression}) \Rightarrow \text{expression}
\]

\[
\sinh(\text{list}) \Rightarrow \text{list}
\]

\(\sinh(\text{expression})\) returns the hyperbolic sine of the argument as an expression.

\(\sinh(\text{list})\) returns a list of the hyperbolic sines of each element of \text{list}.

In Radian angle mode and Rectangular complex format mode:
\[
\sinh([1,5;3,4;2,1;6,-2,1]) \quad \text{ENTER}
\]
\[
\begin{bmatrix}
.360.954 & 305.708 & 239.604
\end{bmatrix}
\]
\[
\begin{bmatrix}
352.912 & 233.495 & 193.564
\end{bmatrix}
\]
\[
\begin{bmatrix}
298.632 & 154.599 & 140.251
\end{bmatrix}
\]

\(\sinh^{-1}()\)

\[\text{MATH/Hyperbolic menu}\]

\[
\sinh^{-1}(\text{expression}) \Rightarrow \text{expression}
\]

\[
\sinh^{-1}(\text{list}) \Rightarrow \text{list}
\]

\(\sinh^{-1}(\text{expression})\) returns the inverse hyperbolic sine of the argument as an expression.

\(\sinh^{-1}(\text{list})\) returns a list of the inverse hyperbolic sines of each element of \text{list}.

In Radian angle mode:
\[
\sinh^{-1}(0) \quad \text{ENTER}
\]
\[
0
\]

\[
\sinh^{-1}([0.2, 0.5]) \quad \text{ENTER}
\]
\[
\begin{bmatrix}
0 & 1.487... & \sinh^{-1}(3)
\end{bmatrix}
\]
\[
\sinh^{-1}(\text{squareMatrix1}) \Rightarrow \text{squareMatrix}
\]

Returns the matrix inverse hyperbolic sine of \text{squareMatrix1}. This is not the same as calculating the inverse hyperbolic sine of each element. For information about the calculation method, refer to \text{cos}().

\text{squareMatrix1} must be diagonalizable. The result always contains floating-point numbers.

\text{In Radian angle mode:}

\[
\sinh^{-1}(\begin{bmatrix} 1.5 & 3; 4.2 & 1.6 \end{bmatrix}) = \begin{bmatrix} 0.41 & 2.155; 1.158 & 1.463 \end{bmatrix}
\]

\text{SinReg} \quad \text{MATH/Statistics/Regressions menu}

\text{SinReg list1, list2[, [iterations], [period] [, list3, list4]]}

Calculates the sinusoidal regression and updates all the system statistics variables.

All the lists must have equal dimensions except for \text{list3}.

\text{list1} represents \text{xlist}.
\text{list2} represents \text{ylist}.
\text{list3} represents category codes.
\text{list4} represents category include list.

\text{iterations} specifies the maximum number of times (1 through 16) a solution will be attempted. If omitted, 8 is used. Typically, larger values result in better accuracy but longer execution times, and vice versa.

\text{period} specifies an estimated period. If omitted, the difference between values in \text{list1} should be equal and in sequential order. If you specify \text{period}, the differences between \text{x} values can be unequal.

\text{Note:} \text{list1} through \text{list3} must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). \text{list4} does not have to be a variable name and cannot be c1–c99.

The output of \text{SinReg} is always in radians, regardless of the angle mode setting.

\text{solve()} \quad \text{MATH/Algebra menu}

\text{solve(equation, var) \Rightarrow Boolean expression}

Returns candidate real solutions of an equation or an inequality for \text{var}. The goal is to return candidates for all solutions. However, there might be equations or inequalities for which the number of solutions is infinite.

Solution candidates might not be real finite solutions for some combinations of values for undefined variables.

For the AUTO setting of the Exact/Approx mode, the goal is to produce exact solutions when they are concise, and supplemented by iterative searches with approximate arithmetic when exact solutions are impractical.
Appendix A: Functions and Instructions

Due to default cancellation of the greatest common divisor from the numerator and denominator of ratios, solutions might be solutions only in the limit from one or both sides.

For inequalities of types ≥, ≤, <, or >, explicit solutions are unlikely unless the inequality is linear and contains only var.

For the EXACT setting of the Exact/Approx mode, portions that cannot be solved are returned as an implicit equation or inequality.

Use the "|" operator to restrict the solution interval and/or other variables that occur in the equation or inequality. When you find a solution in one interval, you can use the inequality operators to exclude that interval from subsequent searches.

false is returned when no real solutions are found. true is returned if solve() can determine that any finite real value of var satisfies the equation or inequality.

Since solve() always returns a Boolean result, you can use "and," "or," and "not" to combine results from solve() with each other or with other Boolean expressions.

Solutions might contain a unique new undefined variable of the form @nj with j being an integer in the interval 1–255. Such variables designate an arbitrary integer.

In real mode, fractional powers having odd denominators denote only the real branch. Otherwise, multiple branched expressions such as fractional powers, logarithms, and inverse trigonometric functions denote only the principal branch. Consequently, solve() produces only solutions corresponding to that one real or principal branch.

Note: See also cSolve(), cZeros(), nSolve(), and zeros().

solve(equation1 and equation2 and ..., {varOrGuess1, varOrGuess2, ..., }) ⇒ Boolean expression

Returns candidate real solutions to the simultaneous algebraic equations, where each varOrGuess specifies a variable that you want to solve for.

Optionally, you can specify an initial guess for a variable. Each varOrGuess must have the form:

variable
  − or −
variable = real or non-real number

For example, x is valid and so is x=3.

solve(x+1)/(x-1)+x=3
	⇒ 2x-2
solve(entry(1)=0,x)

test(ans(1) entry(2))

undefined

limit(entry(3), x=1)

0

solve(x-2 ≥ 2x, x)

x ≥ 2/3

For inequalities of types ≥, ≤, <, or >, explicit solutions are unlikely unless the inequality is linear and contains only var.

solve(5x^2 ≥ x, x)

x ≥ 2/3

solve(entry(1)=0,x)
	x = 1
entry(2)|ans(1)

undefined

limit(entry(3), x=1)

0

solve(x+1)(x-1)/(x-1)+x=3

test(ans(1) entry(2))

undefined

limit(entry(3), x=1)

0

solve(1≤x^2 ≤ 2, x)

x ≥ 2/3

solve(entry(1)=0,x)
	x = 1
entry(2)|ans(1)

undefined

limit(entry(3), x=1)

0

solve(x^2/x ≥ 2, x)

x ≥ 2/3

solve(entry(1)=0,x)
	x = 1
entry(2)|ans(1)

undefined

limit(entry(3), x=1)

0

solve(x=0, x)

false

solve(x=x, x)

true

Since solve() always returns a Boolean result, you can use "and," "or," and "not" to combine results from solve() with each other or with other Boolean expressions.

Solutions might contain a unique new undefined variable of the form @nj with j being an integer in the interval 1–255. Such variables designate an arbitrary integer.

In real mode, fractional powers having odd denominators denote only the real branch. Otherwise, multiple branched expressions such as fractional powers, logarithms, and inverse trigonometric functions denote only the principal branch. Consequently, solve() produces only solutions corresponding to that one real or principal branch.

Note: See also cSolve(), cZeros(), nSolve(), and zeros().
If all of the equations are polynomials and if you do NOT specify any initial guesses, `solve()` uses the lexical Gröbner/Buchberger elimination method to attempt to determine all real solutions.

For example, suppose you have a circle of radius \( r \) at the origin and another circle of radius \( r \) centered where the first circle crosses the positive \( x \)-axis. Use `solve()` to find the intersections.

As illustrated by \( r \) in the example to the right, simultaneous polynomial equations can have extra variables that have no values, but represent given numeric values that could be substituted later.

You can also (or instead) include solution variables that do not appear in the equations. For example, you can include \( z \) as a solution variable to extend the previous example to two parallel intersecting cylinders of radius \( r \).

The cylinder solutions illustrate how families of solutions might contain arbitrary constants of the form \( \alpha_k \), where \( \alpha \) is an integer suffix from 1 through 255. The suffix resets to 1 when you use `ClrHome` or `8:Clear Home`.

For polynomial systems, computation time or memory exhaustion may depend strongly on the order in which you list solution variables. If your initial choice exhausts memory or your patience, try rearranging the variables in the equations and/or `varOrGuess` list.

If you do not include any guesses and if any equation is non-polynomial in any variable but all equations are linear in the solution variables, `solve()` uses Gaussian elimination to attempt to determine all real solutions.

If a system is neither polynomial in all of its variables nor linear in its solution variables, `solve()` determines at most one solution using an approximate iterative method. To do so, the number of solution variables must equal the number of equations, and all other variables in the equations must simplify to numbers.

`solve(x^2+y^2=r^2 and (x-r)^2+y^2=r^2,\{x,y\})` ENTER
\[
x = \frac{r}{2} \quad \text{and} \quad y = \frac{\sqrt{3}r}{2}
\]
or \[
x = \frac{r}{2} \quad \text{and} \quad y = -\frac{\sqrt{3}r}{2}
\]

`solve(x^2+y^2=r^2 and (x-r)^2+y^2=r^2,\{x,y,z\})` ENTER
\[
x = \frac{r}{2} \quad \text{and} \quad y = \frac{\sqrt{3}r}{2} \quad \text{and} \quad z = \alpha_1
\]
or \[
x = \frac{r}{2} \quad \text{and} \quad y = -\frac{\sqrt{3}r}{2} \quad \text{and} \quad z = \alpha_1
\]

`solve(x+e^z*y=1 and x-y=sin(z),\{x,y\})` ENTER
\[
x = \frac{e^z*\sin(z)+1}{e^z+1} \quad \text{and} \quad y = \frac{-(\sin(z)-1)}{e^z+1}
\]

`solve(e^z*y=1 and -y=sin(z),\{y,z\})` ENTER
\[
y = .041... \quad \text{and} \quad z = 3.183...
\]

---

880 Appendix A: Functions and Instructions
Each solution variable starts at its guessed value if there is one; otherwise, it starts at 0.0.

Use guesses to seek additional solutions one by one. For convergence, a guess may have to be rather close to a solution.

```
solve(e^(z) x \geq 1 and \
- y = \sin(z), \{ y, z=2\pi \}) \] \]
\]
\]
```

SortA  MATH/List menu

```
SortA formName[, formName][, formName] ...
SortA vectorName[, vectorName][, vectorName] ...
SortA listName[, listName][, listName] ...

Sorts the elements of the first argument in ascending order.

If you include additional arguments, sorts the elements of each so that their new positions match the new positions of the elements in the first argument.

All arguments must be names of lists or vectors.
All arguments must have equal dimensions.
```

```
\]
\]
```

SortD  MATH/List menu

```
SortD formName[, formName][, formName] ...
SortD vectorName[, vectorName][, vectorName] ...
SortD listName[, listName][, listName] ...

Identical to SortA, except SortD sorts the elements in descending order.
```

```
```

Sphere  MATH/Matrix/Vector ops menu

```
\]
\]
```

```
```

startTmr()  CATALOG

```
startTmr() \] \] \]
```

```
```

```
```

Note: See also checkTmr() and timeCnv().
### stdDev() \ MATH/Statistics menu

**Syntax:**

\[
\text{stdDev(list, freqlist)} \Rightarrow \text{expression}
\]

- **stdDev({a, b, c}) [ENTER]**
- **stdDev({1, 2.5, -6.3, -2}) [ENTER]**

**Description:**

Returns the standard deviation of the elements in `list`.

Each `freqlist` element counts the number of consecutive occurrences of the corresponding element in `list`.

**Note:** `list` must have at least two elements.

**Example:**

\[
\text{stdDev({1.3, 2.5, -6.4}, [3, 2, 5]) [ENTER]}
\]

stdDev({1.3, 2.5, -6.4}, [3, 2, 5])

**Value:**

4.33345

### stdDevPop() \ MATH/Statistics menu

**Syntax:**

\[
\text{stdDevPop(list, freqlist)} \Rightarrow \text{expression}
\]

- **stdDevPop({a, b, c}) [ENTER]**
- **stdDevPop({1, 2.5, -3.0, 1.5, -6.3, -2}) [ENTER]**

**Description:**

Returns the population standard deviation of the elements in `list`.

Each `freqlist` element counts the number of consecutive occurrences of the corresponding element in `list`.

**Note:** `list` must have at least two elements.

**Example:**

In Radian angle and auto modes:

\[
\text{stdDevPop({1.3, 2.5, -6.4}, [3, 2, 5]) [ENTER]}
\]

**Value:**

stdDevPop({1.3, 2.5, -6.4}, [3, 2, 5]) [ENTER]

**Value:**

4.33345

### stdDev(matrix[, freqmatrix]) \ MATH/Statistics menu

**Syntax:**

\[
\text{stdDev(matrix[, freqmatrix])} \Rightarrow \text{matrix}
\]

- **stdDev([1, 2, 5]; [3, 2, 5]) [ENTER]**
- **stdDev([1, 2, 5, 3, 2, 5, 3, 6, -4]; [4, 2, 3, 3, 1, 7]) [ENTER]**

**Description:**

Returns a row vector of the standard deviations of the columns in `matrix`.

Each `freqmatrix` element counts the number of consecutive occurrences of the corresponding element in `matrix`.

**Note:** `matrix` must have at least two rows.

**Example:**

\[
\text{stdDevPop([1, 2, 5], [3, 2, 5]) [ENTER]}
\]

**Value:**

stdDevPop([1, 2, 5, 3, 2, 5, 3, 6, -4]; [4, 2, 3, 3, 1, 7]) [ENTER]

**Value:**

stdDevPop([1, 2, 5, 3, 2, 5, 3, 6, -4]; [4, 2, 3, 3, 1, 7]) [ENTER]

**Value:**

stdDevPop([1, 2, 5, 3, 2, 5, 3, 6, -4]; [4, 2, 3, 3, 1, 7]) [ENTER]
Appendix A: Functions and Instructions

StoGDB  CATALOG

StoGDB  GDBvar

Creates a Graph database (GDB) variable that contains the current:
* Graphing mode
* Y= functions
* Window variables
* Graph format settings
  1- or 2-Graph setting (split screen and ratio settings if 2-Graph mode)
  Angle mode
  Real/complex mode
* Initial conditions if Sequence or Diff Equations mode
* Table flags
* tblStart, tbl, tblInput

You can use RclGDB  GDBvar to restore the graph environment.

*Note:* These items are saved for both graphs in 2-Graph mode.

Stop  CATALOG

Stop

Used as a program instruction to stop program execution.

Program segment:

\[
\begin{align*}
\text{For } i = 1, 10, 1 \\
\text{If } i = 5 \\
\text{Stop} \\
\text{EndFor}
\end{align*}
\]

StoPic  CATALOG

StoPic  picVar, pxlRow, pxlCol, width, height

Displays the graph screen and copies a rectangular area of the display to the variable picVar.

pxlRow and pxlCol, if included, specify the upper-left corner of the area to copy (defaults are 0, 0).

width and height, if included, specify the dimensions, in pixels, of the area. Defaults are the width and height, in pixels, of the current graph screen.

Store  See (store), page 912.

string()  MATH/String menu

string(expression)  \Rightarrow  string

Simplifies expression and returns the result as a character string.

\[
\begin{align*}
\text{string}(1.2345) \hspace{1cm} & \text{\color{red}{ENTER}} \\
& \text{\textcolor{red}{"1.2345"}} \\
\text{string}(1+2) \hspace{1cm} & \text{\color{red}{ENTER}} \\
& \text{\textcolor{red}{"3"}} \\
\text{string}(\cos(x)+\sqrt(3)) \hspace{1cm} & \text{\color{red}{ENTER}} \\
& \text{\textcolor{red}{"\cos(x) + \sqrt(3)"}}
\end{align*}
\]
Style Catalog

**Style equanum, stylePropertyString**

Sets the system graphing function equanum in the current graph mode to use the graphing property stylePropertyString.

*equanum* must be an integer from 1–99 and the function must already exist.

*stylePropertyString* must be one of: "Line", "Dot", "Square", "Thick", "Animate", "Path", "Above", or "Below".

**Note:** In parametric graphing, only the xt half of the pair contains the style information.

Valid style names vs. graphing mode:

- Function: all styles
- Parametric/Polar: line, dot, square, thick, animate, path
- Sequence: line, dot, square, thick
- 3D: none
- Diff Equations: line, dot, square, thick, animate, path

**Note:** Capitalization and blank spaces are optional when entering *stylePropertyString* names.

---

**subMat()** Catalog

```
subMat(matrix1, startRow, startCol, endRow, endCol) ⇒ matrix
```

Returns the specified submatrix of matrix1.

Defaults: startRow=1, startCol=1, endRow=last row, endCol=last column.

**Example:**
```
[1,2,3;4,5,6;7,8,9]
```

```
subMat(m1,2,1,3,2)
```

```
[4 5 7 8]
```

**Example:**
```
sum() MATH/List menu
```

```
sum(list, start, end) ⇒ expression
```

Returns the sum of the elements in list.

Start and end are optional. They specify a range of elements.

```
sum({1,2,3,4,5}) ⇒ 15
```

```
sum((a,2a,3a)) ⇒ 6a
```

```
sum(seq(n,n,1,10)) ⇒ 55
```

```
sum({1,3,5,7,9},3) ⇒ 21
```

```
sum(matrix, start, end) ⇒ matrix
```

Returns a row vector containing the sums of the elements in the columns in matrix.

Start and end are optional. They specify a range of rows.

```
sum([1,2,3;4,5,6;7,8,9],2,3) ⇒ [11,13,15]
```

---

884 Appendix A: Functions and Instructions
switch()  CATALOG

switch(integer) ⇒ integer

Returns the number of the active window. Also can set the active window.

**Note:** Window 1 is left or top; Window 2 is right or bottom.

- If integer = 0, returns the active window number.
- If integer = 1, activates window 1 and returns the previously active window number.
- If integer = 2, activates window 2 and returns the previously active window number.
- If integer is omitted, switches windows and returns the previously active window number.

integer is ignored if the TI-89 Titanium/Voyage™ 200 is not displaying a split screen.

T (transpose)  MATH/Matrix menu

matrixT ⇒ matrix

Returns the complex conjugate transpose of matrix.

\[
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{bmatrix}
\]

\(\text{mat1 ENTER}\)

\[
\begin{bmatrix}
1 & 4 & 7 \\
2 & 5 & 8 \\
3 & 6 & 9 \\
\end{bmatrix}
\]

\(\text{mat1 ENTER}\)

\[
\begin{bmatrix}
a & b \\
c & d \\
\end{bmatrix}
\]

\(\text{mat2 ENTER}\)

\[
\begin{bmatrix}
a & c \\
b & d \\
\end{bmatrix}
\]

\(\text{mat2 ENTER}\)

\[
\begin{bmatrix}
1+i, 2+i, 3+i, 4+i \\
\end{bmatrix}
\]

\(\text{mat3 ENTER}\)

\[
\begin{bmatrix}
1-i, 3-i, 5-i \\
2-i, 4-i, 6-i \\
\end{bmatrix}
\]

\(\text{mat3 ENTER}\)
Table CATALOG

Table expression1, expression2 [, var]

Builds a table of the specified expressions or functions.

The expressions in the table can also be graphed. Expressions entered using the Table or Graph commands are assigned increasing function numbers starting with 1. The expressions can be modified or individually deleted using the edit functions available when the table is displayed by pressing [F2] Header. The currently selected functions in the Y= Editor are temporarily ignored.

To clear the functions created by Table or Graph, execute the ClrGraph command or display the Y= Editor.

If the var parameter is omitted, the current graph-mode independent variable is assumed. Some valid variations of this instruction are:

Function graphing: Table exp1, x
Parametric graphing: Table xExp1, yExp2, t
Polar graphing: Table Exp1, θ

Note: The Table command is not valid for 3D, sequence, or diff equations graphing. An alternative, you may want to use BldData.

\[ \text{In function graphing mode.} \]

\[ \text{Table cos(time), time [ENTER]} \]

\[ \begin{array}{ccc}
1 & 0 & 0 \\
2 & 1.04 & 1.04 \\
3 & 0 & 0 \\
\end{array} \]

\[ \text{Table } \cos(\theta), \theta \text{ [ENTER]} \]

\[ \begin{array}{ccc}
1 & 0 & 1 \\
2 & 1.04 & 1.04 \\
3 & 0 & 0 \\
\end{array} \]

\[ \text{In Degree angle mode:} \]

\[ \tan((\pi/4)^r) \text{ [ENTER]} \]

\[ 1 \]

\[ \tan(45) \text{ [ENTER]} \]

\[ 1 \]

\[ \tan(0, 0.90) \text{ [ENTER]} \]

\[ 0, \sqrt{3}, \text{ undef} \]

\[ \text{In Gradian angle mode:} \]

\[ \tan((\pi/4)^\circ) \text{ [ENTER]} \]

\[ 1 \]

\[ \tan(50) \text{ [ENTER]} \]

\[ 1 \]

\[ \tan(0, 50.10) \text{ [ENTER]} \]

\[ 0, 1, \text{ undef} \]

\[ \text{In Radian angle mode:} \]

\[ \tan(\pi/4) \text{ [ENTER]} \]

\[ 1 \]

\[ \tan(45^\circ) \text{ [ENTER]} \]

\[ 1 \]

\[ \tan((\pi', \pi/3', -\pi, \pi/4)) \text{ [ENTER]} \]

\[ 0, \sqrt{3}, 0, 1 \]
\[ \tan(\text{squareMatrix1}) \Rightarrow \text{squareMatrix} \]

Returns the matrix tangent of \text{squareMatrix1}. This is not the same as calculating the tangent of each element. For information about the calculation method, refer to \text{cos()}. \text{squareMatrix1} must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode:
\[ \tan(\begin{bmatrix} 1 & 2.16; -2.11 \end{bmatrix}) \]
\[ \begin{bmatrix} -28.291... & 26.088... & 11.114... \\ 12.117... & -7.835... & -5.481... \\ 36.818... & -32.806... & -10.459... \end{bmatrix} \]

\[ \tan^{-1}() \]

\[ \tan^{-1}(\text{expression}) \Rightarrow \text{expression} \]
\[ \tan^{-1}(\text{list}) \Rightarrow \text{list} \]

\[ \tan^{-1}(\text{expression}) \] returns the angle whose tangent is \text{expression} as an expression.

\[ \tan^{-1}(\text{list}) \] returns a list of the inverse tangents of each element of \text{list}.

\textbf{Note:} The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

In Radian angle mode:
\[ \tan^{-1}(\begin{bmatrix} 1,5,3;4,2,1;6,\sqrt{2},1 \end{bmatrix}) \]
\[ \begin{bmatrix} .083... & .933... & .425... \\ .748... & .630... & -.070... \\ 1.686... & 1.182... & .455... \end{bmatrix} \]

\[ \tanh() \]

\[ \tanh(\text{expression}) \Rightarrow \text{expression} \]
\[ \tanh(\text{list}) \Rightarrow \text{list} \]

\[ \tanh(\text{expression}) \] returns the hyperbolic tangent of the argument as an expression.

\[ \tanh(\text{list}) \] returns a list of the hyperbolic tangents of each element of \text{list}.

In Radian angle mode:
\[ \tanh(\begin{bmatrix} 1.5,3.4;2.1;6,-2.11 \end{bmatrix}) \]
\[ \begin{bmatrix} -.097... & .933... & .425... \\ .488... & .538... & -.129... \\ 1.282... & -1.034... & .428... \end{bmatrix} \]
Appendix A: Functions and Instructions

MATH/Hyperbolic menu

\[ \tanh^{-1}(\text{expression}) \Rightarrow \text{expression} \]

\[ \tanh^{-1}(\text{list1}) \Rightarrow \text{list} \]

\[ \tanh^{-1}(\text{expression}) \text{ returns the inverse hyperbolic tangent of the argument as an expression.} \]

\[ \tanh^{-1}(\text{list1}) \text{ returns a list of the inverse hyperbolic tangents of each element of list1.} \]

In rectangular complex format mode:

\[ \tanh^{-1}(0) \}
\[ \{ \infty \} \]

\[ \tanh^{-1}(\{1,2,1.3\}) \}
\[ \{ 0.518...\cdot 1.570... \cdot \frac{\ln(2)}{2} \cdot \} \]

\[ \tanh^{-1}(\text{squareMatrix1}) \Rightarrow \text{squareMatrix} \]

Returns the matrix inverse hyperbolic tangent of squareMatrix1. This is not the same as calculating the inverse hyperbolic tangent of each element. For information about the calculation method, refer to \( \cos() \).

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode and Rectangular complex format mode:

\[ \tanh^{-1}(\{1,5,3;4,2,1;6,\pi,2,1\}) \]

\[ \begin{bmatrix} \cdot & 0.099... & 0.164... & 0.267... & -1.490... \cdot \cdot & -1.087... & 0.725... & 0.479... \cdot \cdot \\
\cdot & -0.987... & -0.725... & -1.479... & 0.947... \cdot \cdot \\
\cdot & 0.511... & 2.083... & 1.790... \cdot \cdot \end{bmatrix} \]

MATH/Calculus menu

\[ \text{taylor(expression1, var, order, point)} \Rightarrow \text{expression} \]

Returns the requested Taylor polynomial. The polynomial includes non-zero terms of integer degrees from zero through order in \( \text{var minus point} \). \text{taylor()} returns itself if there is no truncated power series of this order, or if it would require negative or fractional exponents. Use substitution and/or temporary multiplication by a power of \( \text{var minus point} \) to determine more general power series.

point defaults to zero and is the expansion point.

Sometimes \text{tCollect()} will accomplish your goals when the default trigonometric simplification does not. \text{tCollect()} tends to reverse transformations done by \text{tExpand()}. Sometimes applying \text{tExpand()} to a result from \text{tCollect()}, or vice versa, in two separate steps simplifies an expression.

\[ \text{tCollect((cos(\alpha))}^2 \Rightarrow \text{cos(2 \cdot \alpha)} = \frac{1}{2} \]

\[ \text{tCollect(sin(\alpha)cos(\beta)) \Rightarrow sin(\alpha - \beta)+sin(\alpha+\beta)} = \frac{1}{2} \]

\[ \text{tCollect((cos(\alpha))^2 \Rightarrow \cos(2 \cdot \alpha)} = \frac{1}{2} \]

\[ \text{tCollect(sin(\alpha)cos(\beta)) \Rightarrow sin(\alpha - \beta)+sin(\alpha+\beta)} = \frac{1}{2} \]
tExpand() MATH\Algebra\Trig menu

```
tExpand(expression) ⇒ expression
```

Returns an expression in which sines and cosines of integer-multiple angles, angle sums, and angle differences are expanded. Because of the identity \((\sin(x))^2+(\cos(x))^2=1\), there are many possible equivalent results. Consequently, a result might differ from a result shown in other publications.

Sometimes `tExpand()` will accomplish your goals when the default trigonometric simplification does not. `tExpand()` tends to reverse transformations done by `tCollect()`. Sometimes applying `tCollect()` to a result from `tExpand()`, or vice versa, in two separate steps simplifies an expression.

**Note:** Degree-mode scaling by \(\pi/180\) interferes with the ability of `tExpand()` to recognize expandable forms. For best results, `tExpand()` should be used in Radian mode.

```
tExpand(sin(3\(\theta\))) \Rightarrow sin(\(\theta\)) \cdot (cos(\(\theta\)))^3
```

```
tExpand(cos(\(\alpha-\beta\))) \Rightarrow \cos(\(\alpha\)) \cdot \cos(\(\beta\)) + \sin(\(\alpha\)) \cdot \sin(\(\beta\))
```

---

tExpand(sin(3\(\phi\))) \rightarrow 4 \cdot sin(\(\phi\)) \cdot (cos(\(\phi\)))^2 - sin(\(\phi\))

tExpand(cos(\(\alpha-\beta\))) \rightarrow \cos(\(\alpha\)) \cdot \cos(\(\beta\)) + \sin(\(\alpha\)) \cdot \sin(\(\beta\))

---

Text CATALOG

```
Text promptString
```

Displays the character string `promptString` dialog box.

If used as part of a `Dialog...EndDlog` block, `promptString` is displayed inside that dialog box. If used as a standalone instruction, `Text` creates a dialog box to display the string.

```
Text "Have a nice day."
```

Done

---

Then

See If, page 830.

---

timeCnv() CATALOG

```
timeCnv(seconds) ⇒ list
```

Converts seconds to units of time that can be more easily understood for evaluation. The list is in \{days, hours, minutes, seconds\} format.

**Note:** See also `checkTmr()` and `startTmr()`.

```
timeCnv(152442117)
```

\{1764 9 1 57\}

---

Title CATALOG

```
Title titleString, \{Lbl\}
```

Creates the title of a pull-down menu or dialog box when used inside a `Toolbar` or `Custom` construct, or a `Dialog...EndDlog` block.

**Note:** \{Lbl\} is only valid in the `Toolbar` construct. When present, it allows the menu choice to branch to a specified label inside the program.

Program segment:

```
:Dialog
:Title "This is a dialog box"
:Request "Your name",Str1
:Dropdown "Month you were born", seq(string(i), i, 1, 12), Var1
:EndDlog
```

---

Appendix A: Functions and Instructions 889
tmpCnv() CATALOG

tmpCnv(expression1, "tempUnit1", "tempUnit2") ⇒ expression "tempUnit2"

Converts a temperature value specified by expression1 from one unit to another. Valid temperature units are:
- °C Celsius
- °F Fahrenheit
- °K Kelvin
- °R Rankine


For example, 100 °C converts to 212 °F:

0 °C

100 °C

32 °F

212 °F

To convert a temperature range, use ΔtmpCnv() instead.

ΔtmpCnv() CATALOG

ΔtmpCnv(expression1, "tempUnit1", "tempUnit2") ⇒ expression "tempUnit2"

Converts a temperature range (the difference between two temperature values) specified by expression1 from one unit to another. Valid temperature units are:
- °C Celsius
- °F Fahrenheit
- °K Kelvin
- °R Rankine


1 °C and 1 °K have the same magnitude, as do 1 °F and 1 °R. However, 1 °C is 9/5 as large as 1 °F.

For example, a 100 °C range (from 0 °C to 100 °C) is equivalent to a 180 °F range:

0 °C

100 °C

32 °F

212 °F

To convert a particular temperature value instead of a range, use tmpCnv().
### Toolbar

<table>
<thead>
<tr>
<th>Block Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toolbar block EndBar</td>
<td>Creates a toolbar menu. <code>block</code> can be either a single statement or a sequence of statements separated with the <code>:</code> character. The statements can be either Title or Item. Items must have labels. A Title must also have a label if it does not have an Item.</td>
</tr>
</tbody>
</table>

Program segment:
```
:Toolbar
 :  Title "Examples"
 :  Item "Trig", t
 :  Item "Calc", c
 :  Item "Stop", Pexit
 :EndBar
```

**Note:** When run in a program, this segment creates a menu with three choices that branch to three places in the program.

### Trace

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace</td>
<td>Draws a Smart Graph and places the trace cursor on the first defined Y= function at the previously defined cursor position, or at the reset position if regraphing was necessary. Allows operation of the cursor and most keys when editing coordinate values. Several keys, such as the function keys, [APP], and [MODE], are not activated during trace.</td>
</tr>
</tbody>
</table>

**Note:** Press [ENTER] to resume operation.

### Try

<table>
<thead>
<tr>
<th>Block Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Try block1 Else block2 EndTry</td>
<td>Executes <code>block1</code> unless an error occurs. Program execution transfers to <code>block2</code> if an error occurs in <code>block1</code>. Variable <code>errornum</code> contains the error number to allow the program to perform error recovery. <code>block1</code> and <code>block2</code> can be either a single statement or a series of statements separated with the <code>:</code> character.</td>
</tr>
</tbody>
</table>

Program segment:
```
:Try
 :  NewFold(temp)
 :Else
 : Already exists
 :  ClrErr
 :EndTry
```

**Note:** See [ClrErr] and [PassErr].
TwoVar **MATH/Statistics menu**

**TwoVar** \[, list2 [, list3, list4, list5] \]

Calculates the TwoVar statistics and updates all the system statistics variables.

All the lists must have equal dimensions except for list5.

*list1* represents xlist.

*list2* represents ylist.

*list3* represents frequency.

*list4* represents category codes.

*list5* represents category include list.

**Note:** *list1* through *list4* must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). *list5* does not have to be a variable name and cannot be c1–c99.

```
{0,1.2,3.4,5,6} \( \rightarrow \) L1 \( \rightarrow \) Enter
{0,1,2,3,6} \( \rightarrow \) L2 \( \rightarrow \) Enter

TwoVar L1, L2 \( \rightarrow \) Enter  Done

ShowStat \( \rightarrow \) Enter
```

Unarchiv **CATALOG**

**Unarchiv** var1 [, var2 [, var3] ...]

Moves the specified variables from the user data archive memory to RAM.

You can access an archived variable the same as you would a variable in RAM. However, you cannot delete, rename, or store to an archived variable because it is locked automatically.

To archive variables, use **Archive**.

```
10 \( \rightarrow \) arctest \( \rightarrow \) Enter  10
Archive arctest \( \rightarrow \) Enter  Done
5 \( \rightarrow \) arctest \( \rightarrow \) Enter  50
15 \( \rightarrow \) arctest \( \rightarrow \) Enter  Done

Unarchiv arctest \( \rightarrow \) Enter  Done
15 \( \rightarrow \) arctest \( \rightarrow \) Enter  15
```

unitV() **MATH/Matrix/Vector ops menu**

**unitV**(vector1) ⇒ vector

Returns either a row- or column-unit vector, depending on the form of vector1.

*vector1* must be either a single-row matrix or a single-column matrix.

```
unitV([a,b,c]) \( \rightarrow \) Enter
\[
\begin{bmatrix}
a & b & c
\end{bmatrix}
\]

unitV([1,2,1]) \( \rightarrow \) Enter
\[
\begin{bmatrix}
\sqrt{6}/6 & \sqrt{6}/6 & \sqrt{6}/6
\end{bmatrix}
\]

unitV([1;2;3]) \( \rightarrow \) Enter
\[
\begin{bmatrix}
\sqrt{14}/14 & \sqrt{14}/14 & \sqrt{14}/14
\end{bmatrix}
\]
```

Unlock **CATALOG**

**Unlock** var1 [, var2 [, var3] ...]

Unlocks the specified variables.

**Note:** The variables can be locked using the **Lock** command.

892  **Appendix A: Functions and Instructions**
Appendix A: Functions and Instructions

### Variance

**Syntax:**

```
variance(list[, freqlist]) ⇒ expression
```

**Description:**

Returns the variance of `list`.

Each `freqlist` element counts the number of consecutive occurrences of the corresponding element in `list`.

**Note:** `list` must contain at least two elements.

**Examples:**

- `variance({a,b,c})⇒a^2-a⋅(b+c)+b^2-b⋅c+c^2/3`
- `variance({1,2,5,6,3,-2})⇒31/2`
- `variance({1,3,5},{4,6,2})⇒68/33`

**Matrix Variance**

```
variance(matrix[, frecmatrix]) ⇒ matrix
```

Returns a row vector containing the variance of each column in `matrix`.

Each `frecmatrix` element counts the number of consecutive occurrences of the corresponding element in `matrix`.

**Note:** `matrix` must contain at least two rows.

**Examples:**

- `variance([1,2,5;3,0,1;.5,.7,3])⇒[4.75,1.03,4]`
- `variance([-1.1,2;2,3.4,5.1;2.3,4.3],[6,3;2,4;5,1])⇒[3.91731,2.08411]`

### When

**Syntax:**

```
when(condition, trueResult[, falseResult], unknownResult) ⇒ expression
```

**Description:**

Returns `trueResult`, `falseResult`, or `unknownResult`, depending on whether `condition` is true, false, or unknown. Returns the input if there are too few arguments to specify the appropriate result.

- Omit both `falseResult` and `unknownResult` to make an expression defined only in the region where `condition` is true.
- Use an `undef` `falseResult` to define an expression that graphs only on an interval.
- Omit only the `unknownResult` to define a two-piece expression.
- Nest `when()` to define expressions that have more than two pieces.

**Examples:**

- `ClrGraph
  Graph when(x<0,x+3)|x=5
  when(x<0,3+x)
  Use an `undef` `falseResult` to define an expression that graphs only on an interval.

- `Graph when(x<0,x+3,5,if x<0,x^2)
  Omit only the `unknownResult` to define a two-piece expression.

- `Nest `when()` to define expressions that have more than two pieces.`
when() is helpful for defining recursive functions.

```
when(n>0,n*factoral(n-1),1)

> factorial(n) [ENTER] Done
factorial(3) [ENTER] 6
3! [ENTER] 6
```

**While**

**CATALOG**

```
While condition
  block
EndWhile
```

Executes the statements in `block` as long as `condition` is true.

`block` can be either a single statement or a sequence of statements separated with the `:` character.

**Program segment:**

```
: 3->i
: 0->temp
:while i<=20
  : temp+1/i->temp
  : i+1->i
:Endwhile
:Disp "sum of reciprocals up to 20",temp
```

**“With”**

See | page 912.

**xor**

**MATH/Test menu**

```
Boolean expression1 xor Boolean expression2  =>  Boolean expression
```

Returns true if `Boolean expression1` is true and `Boolean expression2` is false, or vice versa.

Returns false if `Boolean expression1` and `Boolean expression2` are both true or both false. Returns a simplified Boolean expression if either of the original Boolean expressions cannot be resolved to true or false.

**Note:** See or.

```
inHex base mode:
0h7AC36 xor 0h3D5F [ENTER] 0h79169
```

```
in Bin base mode:
0b100101 xor 0b100 [ENTER] 0b100001
```

**Note:** A binary entry can have up to 32 digits (not counting the 0b prefix). A hexadecimal entry can have up to 8 digits.

An important note: Zero, not the letter O.
**XorPic**  
**CATALOG**

**XorPic** picVar, row [, column]

Displays the picture stored in `picVar` on the current Graph screen.

Uses xor logic for each pixel. Only those pixel positions that are exclusive to either the screen or the picture are turned on. This instruction turns off pixels that are turned on in both images.

`picVar` must contain a pic data type.

`row` and `column`, if included, specify the pixel coordinates for the upper left corner of the picture. Defaults are (0, 0).

**zeros()**  
**MATH/Algebra menu**

`zeros(expression, var) ⇒ list`

Returns a list of candidate real values of `var` that make `expression=0`. `zeros()` does this by computing `expolist(solve(expression=0, var), var)`.

For some purposes, the result form for `zeros()` is more convenient than that of `solve()`. However, the result form of `zeros()` cannot express implicit solutions, solutions that require inequalities, or solutions that do not involve `var`.

**Note:** See also `cSolve()`, `cZeros()`, and `solve()`.

`zeros({expression1, expression2}, {varOrGuess1, varOrGuess2, ...}) ⇒ matrix`

Returns candidate real zeros of the simultaneous algebraic expressions, where each `varOrGuess` specifies an unknown whose value you seek.

Optionally, you can specify an initial guess for a variable. Each `varOrGuess` must have the form:

- `variable`  
- or  
- `variable = real or non-real number`

For example, `x` is valid and so is `x=3`.

If all of the expressions are polynomials and if you do NOT specify any initial guesses, `zeros()` uses the lexical Gröbner/Buchberger elimination method to attempt to determine all real zeros.

For example, suppose you have a circle of radius `r` at the origin and another circle of radius `r` centered where the first circle crosses the positive x-axis. Use `zeros()` to find the intersections.

As illustrated by `r` in the example to the right, simultaneous polynomial expressions can have extra variables that have no values, but represent given numeric values that could be substituted later.

Each row of the resulting matrix represents an alternate zero, with the components ordered the same as the `varOrGuess` list. To extract a row, index the matrix by `[row].`
You can also (or instead) include unknowns that do not appear in the expressions. For example, you can include \( z \) as an unknown to extend the previous example to two parallel intersecting cylinders of radius \( r \). The cylinder zeros illustrate how families of zeros might contain arbitrary constants in the form \( c_k \), where \( k \) is an integer suffix from 1 through 255. The suffix resets to 1 when you use \( \text{ClrHome} \) or \( \text{Clear Home} \).

For polynomial systems, computation time or memory exhaustion may depend strongly on the order in which you list unknowns. If your initial choice exhausts memory or your patience, try rearranging the variables in the expressions and/or \( \text{varOrGuess} \) list.

If you do not include any guesses and if any expression is non-polynomial in any variable but all expressions are linear in the unknowns, \( \text{zeros}() \) uses Gaussian elimination to attempt to determine all real zeros.

If a system is neither polynomial in all of its variables nor linear in its unknowns, \( \text{zeros}() \) determines at most one zero using an approximate iterative method. To do so, the number of unknowns must equal the number of expressions, and all other variables in the expressions must simplify to numbers.

Each unknown starts at its guessed value if there is one; otherwise, it starts at 0.0.

Use guesses to seek additional zeros one by one. For convergence, a guess may have to be rather close to a zero.

**ZoomBox CATALOG**

**ZoomBox**

Displays the Graph screen, lets you draw a box that defines a new viewing window, and updates the window.

In function graphing mode:

\[ 1.25x \cdot \cos(x) \cdot y1(x) \quad \text{Done} \]

\[ \text{ZoomStd:ZoomBox ENTER} \]

The display after defining \( \text{ZoomBox} \) by pressing \( \text{ENTER} \) the second time.

896 Appendix A: Functions and Instructions
**ZoomData**

Adjusts the window settings based on the currently defined plots (and data) so that all statistical data points will be sampled, and displays the Graph screen.

*Note:* Does not adjust $y_{min}$ and $y_{max}$ for histograms.

In function graphing mode:

```
{1, 2, 3, 4} → L1 [ENTER]     {1 2 3 4}
{2, 3, 4, 5} → L2 [ENTER]     {2 3 4 5}
newPlot 1.1.1.1 [ENTER]       Done
ZoomStd [ENTER]
```

**ZoomDec**

Adjusts the viewing window so that $\Delta x$ and $\Delta y = 0.1$ and displays the Graph screen with the origin centered on the screen.

In function graphing mode:

```
1.25x* cos(x) → y1(x) [ENTER]     Done
ZoomStd [ENTER]
```

```
HOME
ZoomDec [ENTER]
```
### ZoomFit  
**Display**

Displays the Graph screen, and calculates the necessary window dimensions for the dependent variables to view all the picture for the current independent variable settings.

**In function graphing mode:**

\[
1.25x \cos(x) \cdot y_1(x) \text{ ENTER} \quad \text{Done}
\]

#### ZoomIn  
**Display**

Displays the Graph screen, lets you set a center point for a zoom in, and updates the viewing window.

The magnitude of the zoom is dependent on the Zoom factors xFact and yFact. In 3D Graph mode, the magnitude is dependent on xFact, yFact, and zFact.

**In function graphing mode:**

\[
1.25x \cos(x) \cdot y_1(x) \text{ ENTER} \quad \text{Done}
\]

#### ZoomInt  
**Display**

Displays the Graph screen, lets you set a center point for the zoom, and adjusts the window settings so that each pixel is an integer in all directions.

**In function graphing mode:**

\[
1.25x \cos(x) \cdot y_1(x) \text{ ENTER} \quad \text{Done}
\]
Appendix A: Functions and Instructions

**ZoomOut**  
**Displays the Graph screen, lets you set a center point for a zoom out, and updates the viewing window.**  
The magnitude of the zoom is dependent on the Zoom factors xFact and yFact. In 3D Graph mode, the magnitude is dependent on xFact, yFact, and zFact.

![ZoomOut diagram]

**ZoomPrev**  
**Displays the Graph screen, and updates the viewing window with the settings in use before the last zoom.**

**ZoomRcl**  
**Displays the Graph screen, and updates the viewing window using the settings stored with the ZoomSto instruction.**

**ZoomSqr**  
**Displays the Graph screen, adjusts the x or y window settings so that each pixel represents an equal width and height in the coordinate system, and updates the viewing window.**  
In 3D Graph mode, ZoomSqr lengthens the shortest two axes to be the same as the longest axis.

![ZoomSqr diagram]
### ZoomStd CATALOG

Sets the window variables to the following standard values, and then updates the viewing window.

**Function graphing:**
x: [−10, 10, 1], y: [−10, 10, 1] and xres=2

**Parametric graphing:**
t: [0, 2π, π/24], x: [−10, 10, 1], y: [−10, 10, 1]

**Polar graphing:**
θ: [0, 2π, π/24], x: [−10, 10, 1], y: [−10, 10, 1]

**Sequence graphing:**
nmin=1, nmax=10, plotStrt=1, plotStep=1,
x: [−10, 10, 1], y: [−10, 10, 1]

**3D graphing:**
eyeθ=20, eyef=70, eyeψ=0
x: [−10, 10, 14], y: [−10, 10, 14],
z: [−10, 10], ncontour=5

**Differential equations graphing:**
t: [0, 10, 0.1, 0], x: [−1, 10, 1], y: [−10, 10, 1], ncurves=0, Estep=1, difftol=0.001, fldres=14, dtimel=0

---

### ZoomSto CATALOG

Stores the current Window settings in the Zoom memory. You can use ZoomRcl to restore the settings.

---

### ZoomTrig CATALOG

Displays the Graph screen, sets Δx to π/24, and xsc1 to π/2, centers the origin, sets the y settings to [−4, 4, .5], and updates the viewing window.

---

### + (add) key

expression1 + expression2 ⇒ expression

Returns the sum of expression1 and expression2.
| list1 + list2 ⇒ list                                                                 | \{22, π/2\}+L1 ENTER \{22 π/2\} |
| matrix1 + matrix2 ⇒ matrix                                                           | \{10,5,π/2\}+L2 ENTER \{10 5 π/2\} |
|                                                                                     | L1+L2 ENTER \{32 π/5\} |
|                                                                                     | ans(1)+\{π, -5, -π\} ENTER \{π+32 π 0\} |
|                                                                                     | \{a,b,c,d\}+[1,0;0,1] ENTER |
| Returns a list (or matrix) containing the sums of corresponding elements in list1 and list2 (or matrix1 and matrix2). |
| Dimensions of the arguments must be equal.                                          |

| expression + list1 ⇒ list                                                             | 15+\{10,15,20\} ENTER \{25 30 35\} |
| list1 + expression ⇒ list                                                             | \{10,15,20\}+15 ENTER \{25 30 35\} |
|                                                                                     | 20+[1,2;3,4] ENTER |
| Returns a list containing the sums of expression and each element in list1.          |

| expression + matrix1 ⇒ matrix                                                         | 20+[1,2;3,4] ENTER |
| matrix1 + expression ⇒ matrix                                                         | 21 2,2 2, |
|                                                                                     | \[3 24\] |
| Returns a matrix with expression added to each element on the diagonal of matrix1. matrix1 must be square. |

**Note:** Use .+ (dot plus) to add an expression to each element.

| list1 -- list2 ⇒ list                                                                 | \{22,π,π/2\}--L1 ENTER \{12 5 0\} |
| matrix1 -- matrix2 ⇒ matrix                                                           | \{3,4\}--[1,2] ENTER \{2 2\} |
|                                                                                     | Subtracts each element in list2 (or matrix2) from the corresponding element in list1 (or matrix1), and returns the results. |
| Dimensions of the arguments must be equal.                                          |

| expression -- list1 ⇒ list                                                             | 15--\{10,15,20\} ENTER \{5 0 -5\} |
| list1 -- expression ⇒ list                                                             | \{10,15,20\}--15 ENTER \{-5 0 5\} |
|                                                                                     | Subtracts each list1 element from expression or subtracts expression from each list1 element, and returns a list of the results. |

| expression -- matrix1 ⇒ matrix                                                         | 20--[1,2;3,4] ENTER |
| matrix1 -- expression ⇒ matrix                                                         | 19 -2, |
|                                                                                     | \[-3 16\] |
| expression -- matrix1 returns a matrix of expression times the identity matrix minus matrix1. matrix1 must be square. |
| matrix1 -- expression returns a matrix of expression times the identity matrix subtracted from matrix1. matrix1 must be square. |

**Note:** Use .- (dot minus) to subtract an expression from each element.
(multiply) \( \textbf{key} \)

\[
\text{expression1} \ast \text{expression2} \Rightarrow \text{expression}
\]

Returns the product of \text{expression1} and \text{expression2}.

\[
2 \ast 3.45 \{\text{ENTER}\} \quad 6.9
\]

\[
\text{list1} \ast \text{list2} \Rightarrow \text{list}
\]

Returns a list containing the products of the corresponding elements in \text{list1} and \text{list2}.

\[
\{1.0,2.3\} \ast \{4,5.6\} \{\text{ENTER}\} \quad \{4.0,1.8\}
\]

Dimensions of the lists must be equal.

\[
\text{matrix1} \ast \text{matrix2} \Rightarrow \text{matrix}
\]

Returns the matrix product of \text{matrix1} and \text{matrix2}.

The number of rows in \text{matrix1} must equal the number of columns in \text{matrix2}.

\[
\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \ast \begin{bmatrix} 5 \\ 6 \end{bmatrix} \{\text{ENTER}\} \quad \begin{bmatrix} 1 \ast 5 + 2 \ast 6 \\ 3 \ast 5 + 4 \ast 6 \end{bmatrix}
\]

\[
\text{list1} \ast \text{expression} \Rightarrow \text{list}
\]

Returns a list containing the products of \text{expression} and each element in \text{list1}.

\[
\{1.0,2,3\} \ast \{4,5,6\} \{\text{ENTER}\} \quad \{4.0,10,18\}
\]

\[
\text{list1} \ast \text{expression} \Rightarrow \text{list}
\]

Returns a list containing the products of expression and each element in \text{list1}.

\[
p \ast \{4,5,6\} \{\text{ENTER}\} \quad \{4 \ast p, 5 \ast p, 6 \ast p\}
\]

\[
\text{expression} \ast \text{list1} \Rightarrow \text{list}
\]

Returns a list containing the products of \text{expression} divided by each element in \text{list1}.

\[
2 / 3.45 \{\text{ENTER}\} \quad .57971
\]

\[
\text{list1} \ast \text{expression} \Rightarrow \text{list}
\]

Returns a list containing the products of \text{expression} divided by \text{list1}.

\[
\{1.0,2.3\} / \{4,5.6\} \{\text{ENTER}\} \quad \{.25, 5/12\}
\]

Dimensions of the lists must be equal.

\[
\text{expression} / \text{list1} \Rightarrow \text{list}
\]

Returns a list containing the quotients of \text{expression} divided by \text{list1}.

\[
a / \{3,a,\sqrt{a}\} \{\text{ENTER}\} \quad \{\sqrt{a}, 1/3, 1\}
\]

\[
\text{list1} / \text{expression} \Rightarrow \text{list}
\]

Returns a list containing the quotients of \text{expression} divided by \text{list1} or \text{list1} divided by \text{expression}.

\[
\{a,b,c\} / (a \ast b \ast c) \{\text{ENTER}\} \quad \{1/b \cdot c, 1/a \cdot c, 1/a \cdot b\}
\]

\[
\text{matrix1} / \text{expression} \Rightarrow \text{matrix}
\]

Returns a matrix containing the quotients of \text{matrix1} divided by \text{expression}.

\[
\text{Note: Use } / \text{(dot divide)} \text{ to divide an expression by each element.}
\]

\[
\text{expression} / \text{expression} \Rightarrow \text{expression}
\]

Returns the quotient of \text{expression1} divided by \text{expression2}.

\[
2 / 3.45 \{\text{ENTER}\} \quad .57971
\]

\[
\text{list1} / \text{list2} \Rightarrow \text{list}
\]

Returns a list containing the quotients of \text{list1} divided by \text{list2}.

\[
\{1.0,2.3\} / \{4,5.6\} \{\text{ENTER}\} \quad \{.25, 5/12\}
\]

Dimensions of the lists must be equal.

\[
\text{expression} / \text{list1} \Rightarrow \text{list}
\]

Returns a list containing the quotients of \text{expression} divided by \text{list1}.

\[
\{a,b,c\} / (a \ast b \ast c) \{\text{ENTER}\} \quad \{1/b \cdot c, 1/a \cdot c, 1/a \cdot b\}
\]

\[
\text{matrix1} / \text{expression} \Rightarrow \text{matrix}
\]

Returns a matrix containing the quotients of \text{matrix1} divided by \text{expression}.

\[
\text{Note: Use } / \text{(dot divide)} \text{ to divide an expression by each element.}\]
### Appendix A: Functions and Instructions

#### ^ (power)

**expression1 ^ expression2** → expression

Returns the first argument raised to the power of the second argument.

For a list, returns the elements in list1 raised to the power of the corresponding elements in list2.

In the real domain, fractional powers that have reduced exponents with odd denominators use the real branch versus the principal branch for complex mode.

<table>
<thead>
<tr>
<th>expression1 ^ expression2</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>4^2</td>
<td>16</td>
</tr>
<tr>
<td>{a,2,c}^1{b,3}</td>
<td>{a 2^b c^3}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>list ^ list2</th>
<th>list</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a,2,c} ^ {1,b,3}</td>
<td>{a 2^b c^3}</td>
</tr>
</tbody>
</table>

#### list ^ expression → list

Returns expression raised to the power of the elements in list1.

<table>
<thead>
<tr>
<th>list ^ expression</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2,3,4) ^ -2</td>
<td>(1 1/4 1/9 1/16)</td>
</tr>
</tbody>
</table>

#### squareMatrix1 ^ integer → matrix

Returns squareMatrix1 raised to the integer power.

squareMatrix1 must be a square matrix.

If integer = -1, computes the inverse matrix.
If integer < -1, computes the inverse matrix to an appropriate positive power.

<table>
<thead>
<tr>
<th>squareMatrix1 ^ integer</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,2;3,4] ^ -2</td>
<td>[1.1 1/2 1/4 1/16]</td>
</tr>
<tr>
<td>[1,2;3,4] ^ 1</td>
<td>[1 2 3 4]</td>
</tr>
</tbody>
</table>

#### .+ (dot add)

**matrix1 .+ matrix2** → matrix

**expression .+ matrix1** → matrix

matrix1 .+ matrix2 returns a matrix that is the sum of each pair of corresponding elements in matrix1 and matrix2.

expression .+ matrix1 returns a matrix that is the sum of expression and each element in matrix1.

<table>
<thead>
<tr>
<th>matrix1 .+ matrix2</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>[a,b; c, d] .+ [e,f; g, h]</td>
<td>[a+e b+f c+g d+h]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>expression .+ matrix1</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 .+ [1,2;3,4]</td>
<td>[3,4;5,6]</td>
</tr>
<tr>
<td>(x + 3) .+ [1,2;3,4]</td>
<td>[x + 3 1 2 3]</td>
</tr>
</tbody>
</table>

#### .- (dot subt.)

**matrix1 .- matrix2** → matrix

**expression .- matrix1** → matrix

matrix1 .- matrix2 returns a matrix that is the difference between each pair of corresponding elements in matrix1 and matrix2.

expression .- matrix1 returns a matrix that is the difference of expression and each element in matrix1.

<table>
<thead>
<tr>
<th>matrix1 .- matrix2</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>[a,b; c, d] .- [e,f; g, h]</td>
<td>[a-e b-f c-g d-h]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>expression .- matrix1</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 .- [1,2;3,4]</td>
<td>[1 2 3 4]</td>
</tr>
<tr>
<td>(x - 3) .- [1,2;3,4]</td>
<td>[x - 3 1 2 3]</td>
</tr>
</tbody>
</table>
**. (dot mult.)**

**keys**

\[ \begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} \times \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix} \]

- \[ \text{matrix1 \ast matrix2} \Rightarrow \text{matrix} \]
- \[ \text{expression \ast matrix1} \Rightarrow \text{matrix} \]

- \[ \text{matrix1 \ast matrix2} \text{ returns a matrix that is the product of each pair of corresponding elements in matrix1 and matrix2.} \]
- \[ \text{expression \ast matrix1} \text{ returns a matrix containing the products of \text{expression} and each element in matrix1.} \]

**/. (dot divide)**

**keys**

\[ \begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} \div \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix} \]

- \[ \begin{bmatrix} a & b \\ c & d \end{bmatrix} \div \begin{bmatrix} e & f \\ g & h \end{bmatrix} \]

- \[ \text{matrix1 \div matrix2} \Rightarrow \text{matrix} \]
- \[ \text{expression \div matrix1} \Rightarrow \text{matrix} \]

- \[ \text{matrix1 \div matrix2} \text{ returns a matrix that is the quotient of each pair of corresponding elements in matrix1 and matrix2.} \]
- \[ \text{expression \div matrix1} \text{ returns a matrix that is the quotient of \text{expression} and each element in matrix1.} \]

**.^(dot power)**

**keys**

\[ \begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} \times \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix} \]

- \[ \begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} \times \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix} \]

- \[ \text{matrix1 \times matrix2} \Rightarrow \text{matrix} \]
- \[ \text{expression \times matrix1} \Rightarrow \text{matrix} \]

- \[ \text{matrix1 \times matrix2} \text{ returns a matrix where each element in matrix2 is the exponent for the corresponding element in matrix1.} \]
- \[ \text{expression \times matrix1} \text{ returns a matrix where each element in matrix1 is the exponent for \text{expression}.} \]

**~ (negate)**

**key and MATH/Base menu**

- \[ \text{expression1} \Rightarrow \text{expression} \]
- \[ \text{matrix1} \Rightarrow \text{matrix} \]

- \[ \text{expression1} \Rightarrow \text{expression} \]
- \[ \text{matrix1} \Rightarrow \text{matrix} \]

Returns the negation of the argument.

For a list or matrix, returns all the elements negated.

If \text{expression1} is a binary or hexadecimal integer, the negation gives the two’s complement.

**% (percent)**

**CHAR/Punctuation menu**

- \[ \text{expression1} \% \Rightarrow \text{expression} \]
- \[ \text{list1} \% \Rightarrow \text{list} \]
- \[ \text{matrix1} \% \Rightarrow \text{matrix} \]

- \[ \text{expression1} \% \Rightarrow \text{expression} \]
- \[ \text{list1} \% \Rightarrow \text{list} \]
- \[ \text{matrix1} \% \Rightarrow \text{matrix} \]

Returns \[ \frac{100}{\text{argument}} \]

For a list or matrix, returns a list or matrix with each element divided by 100.
Appendix A: Functions and Instructions

\[
= (\text{equal}) \quad \boxed{\text{key}}
\]

expression1 \(=\) expression2 \(\Rightarrow\) Boolean expression
list1 \(=\) list2 \(\Rightarrow\) Boolean list
matrix1 \(=\) matrix2 \(\Rightarrow\) Boolean matrix

Returns true if expression1 is determined to be equal to expression2.
Returns false if expression1 is determined to not be equal to expression2.
Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

Example function listing using math test symbols: =, ≠, <, ≤, ≥

: \(g(x)\)
:Func
:If \(x\leq5\) Then
: Return 5
: ElseIf \(x>5\) and \(x<0\) Then
: Return \(-x\)
: ElseIf \(x\geq0\) and \(x\leq10\) Then
: Return \(x\)
: ElseIf \(x>10\) Then
: Return 3
:EndIf
:EndFunc

Graph \(g(x)\) \[ENTER\]

\[
\neq (\text{not equal}) \quad \boxed{\text{key}}
\]

expression1 \(\neq\) expression2 \(\Rightarrow\) Boolean expression
list1 \(\neq\) list2 \(\Rightarrow\) Boolean list
matrix1 \(\neq\) matrix2 \(\Rightarrow\) Boolean matrix

Returns true if expression1 is determined to be not equal to expression2.
Returns false if expression1 is determined to be equal to expression2.
Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

See "=" (equal) example.

\[
< \quad \boxed{\text{[><] key}}
\]

expression1 \(<\) expression2 \(\Rightarrow\) Boolean expression
list1 \(<\) list2 \(\Rightarrow\) Boolean list
matrix1 \(<\) matrix2 \(\Rightarrow\) Boolean matrix

Returns true if expression1 is determined to be less than expression2.
Returns false if expression1 is determined to be greater than or equal to expression2.
Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

See "=" (equal) example.
≤

expression1 ≤ expression2 ⇒ Boolean expression  
list1 ≤ list2 ⇒ Boolean list  
matrix1 ≤ matrix2 ⇒ Boolean matrix

Returns true if expression1 is determined to be less than or equal to expression2.

Returns false if expression1 is determined to be greater than expression2.

Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

> 

expression1 > expression2 ⇒ Boolean expression  
list1 > list2 ⇒ Boolean list  
matrix1 > matrix2 ⇒ Boolean matrix

Returns true if expression1 is determined to be greater than expression2.

Returns false if expression1 is determined to be less than or equal to expression2.

Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

≥

expression1 ≥ expression2 ⇒ Boolean expression  
list1 ≥ list2 ⇒ Boolean list  
matrix1 ≥ matrix2 ⇒ Boolean matrix

Returns true if expression1 is determined to be greater than or equal to expression2.

Returns false if expression1 is determined to be less than expression2.

Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

!(factorial)

expression1! ⇒ expression  
list1! ⇒ list  
matrix1! ⇒ matrix

Returns the factorial of the argument.

For a list or matrix, returns a list or matrix of factorials of the elements.

The TI-89 computes a numeric value for only non-negative whole-number values.
& (append)  key

\text{string1} \& \text{string2} \Rightarrow \text{string}

Returns a text string that is \text{string2} appended to \text{string1}.

\textbf{J0} (integrate)  key

\text{expression}, \text{var}, \text{lower}, \text{upper} \Rightarrow \text{expression}

\text{list}, \text{var}, \text{order} \Rightarrow \text{list}

\text{matrix}, \text{var}, \text{order} \Rightarrow \text{matrix}

Returns the integral of \text{expression1} with respect to the variable \text{var} from \text{lower} to \text{upper}.

Returns an anti-derivative if \text{lower} and \text{upper} are omitted. A symbolic constant of integration such as \(C\) is omitted.

However, \text{lower} is added as a constant of integration if only \text{upper} is omitted.

Equally valid anti-derivatives might differ by a numeric constant. Such a constant might be disguised—particularly when an anti-derivative contains logarithms or inverse trigonometric functions. Moreover, piecewise constant expressions are sometimes added to make an anti-derivative valid over a larger interval than the usual formula.

\text{expression1} \Rightarrow \text{expression2}

\text{list1}, \text{var}, \text{order} \Rightarrow \text{list2}

\text{matrix1}, \text{var}, \text{order} \Rightarrow \text{matrix2}

\(\frac{d}{dx} \) returns itself for pieces of \text{expression1} that it cannot determine as an explicit finite combination of its built-in functions and operators.

When \text{lower} and \text{upper} are both present, an attempt is made to locate any discontinuities or discontinuous derivatives in the interval \text{lower} < \text{var} < \text{upper} and to subdivide the interval at those places.

For the AUTO setting of the Exact/Approx mode, numerical integration is used where applicable when an anti-derivative or a limit cannot be determined.

For the APPROX setting, numerical integration is tried first, if applicable. Anti-derivatives are sought only where such numerical integration is inapplicable or fails.

\text{J0} can be nested to do multiple integrals.

Integration limits can depend on integration variables outside them.

\textbf{Note:} See also \texttt{nint()}.
\(\sqrt{\text{(expression)}} \Rightarrow \text{expression}\)

\(\sqrt{\text{(list)}} \Rightarrow \text{list}\)

Returns the square root of the argument.

For a list, returns the square roots of all the elements in list.

\[\sqrt{4(\text{enter})} \quad \sqrt{\{(9, a.4)\}(\text{enter})} \quad \{3 \sqrt{a} 2\}\]

\(\Pi()\) (product) MATH/Calculus menu

\(\Pi(\text{expression1}, \text{var}, \text{low}, \text{high}) \Rightarrow \text{expression}\)

Evaluates expression1 for each value of var from low to high, and returns the product of the results.

\[\Pi(1/n.n.1.5) \text{enter} \quad \frac{1}{120}\]

\[\Pi(k*2,k.1.n) \text{enter} \quad (n!)^2\]

\[\Pi\{(l/n.n.2),n.1.5\} \text{enter} \quad \{\frac{1}{120} 120 32\}\]

\[\Pi(\text{expression1}, \text{var}, \text{low} - 1) \Rightarrow 1\]

\[\Pi(k,k.4.3) \text{enter} \quad 1\]

\[\Pi(1/k,k.4.1) \text{enter} \quad 6\]

\[\Pi(1/k,k.4.1) \ast \Pi(1/k,k.2.4) \text{enter} \quad 1/4\]

\(\Sigma()\) (sum) MATH/Calculus menu

\(\Sigma(\text{expression1}, \text{var}, \text{low}, \text{high}) \Rightarrow \text{expression}\)

Evaluates expression1 for each value of var from low to high, and returns the sum of the results.

\[\Sigma(1/n.n.1.5) \text{enter} \quad 137\]

\[\Sigma(k*2,k.1.n) \text{enter} \quad \frac{n \ast (n + 1) \ast (2 \ast n + 1)}{6}\]

\[\Sigma(1/n^2,n.1.\infty) \text{enter} \quad \frac{\pi^2}{6}\]

\[\Sigma(k,k.4.3) \text{enter} \quad 0\]

\[\Sigma(k,k.4.1) \text{enter} \quad -5\]

\[\Sigma(k,k.4.1) \ast \Sigma(k,k.2.4) \text{enter} \quad 4\]

\# (indirection) CATALOG

\# \text{varNameString}

Refers to the variable whose name is \text{varNameString}. This lets you create and modify variables from a program using strings.

Program segment:

\[
\text{\begin{verbatim}
:Request "Enter Your Name".str1
:NewFold \#str1
:"
:For i,1,5,1
: ClrGraph
: Graph i*x
: StoPic \#("pic" & string(i))
:EndFor
:"
\end{verbatim}}
\]

908 Appendix A: Functions and Instructions
### g (gradian)  MATH/ANGLE menu

<table>
<thead>
<tr>
<th>expression</th>
<th>list</th>
<th>matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>$expression$^g$</td>
<td>$expression$</td>
<td>$expression$</td>
</tr>
<tr>
<td>$list^g$</td>
<td>$list$</td>
<td>$list$</td>
</tr>
<tr>
<td>$matrix^g$</td>
<td>$matrix$</td>
<td>$matrix$</td>
</tr>
</tbody>
</table>

This function gives you a way to use a gradian angle while in the Degree or Radian modes.

In Radian angle mode, multiplies $expression$ by $\pi/200$. In Degree angle mode, multiplies $expression$ by $g/100$.

In Gradian mode, returns $expression$ unchanged.

In Degree, Gradian or Radian mode:

$cos(50^g) \quad \text{ENTER} \quad \sqrt{2}$

$cos((0, 100^g, 200^g)) \quad \text{ENTER} \quad \{1.0 \cdot -1\}$

### f (radian)  MATH/ANGLE menu

<table>
<thead>
<tr>
<th>expression</th>
<th>list</th>
<th>matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>$expression^f$</td>
<td>$list^f$</td>
<td>$matrix^f$</td>
</tr>
</tbody>
</table>

In Degree angle mode, multiplies $expression$ by $180/\pi$. In Radian angle mode, returns $expression$ unchanged. In Gradian mode, multiplies $expression$ by $100/\pi$.

This function gives you a way to use a radian angle while in Degree and Gradian mode.

**Hint:** Use $^f$ if you want to force radians in a function or program definition regardless of the mode that prevails when the function or program is used.

In Degree, Gradian or Radian angle mode:

$cos((\pi/4)^f) \quad \text{ENTER} \quad \sqrt{2}$

$cos((0^f \cdot (\pi/12)^f, -\pi^f)) \quad \text{ENTER} \quad \{1 \cdot \sqrt{3} \cdot 4, -1\}$

### o (degree)  $\text{2}^\text{nd} \ [^\circ]$ key

<table>
<thead>
<tr>
<th>expression</th>
<th>list</th>
<th>matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>$expression^o$</td>
<td>$list^o$</td>
<td>$matrix^o$</td>
</tr>
</tbody>
</table>

In Radian angle mode, multiplies $expression$ by $\pi/180$. In Degree angle mode, returns $expression$ unchanged. In Gradian mode, multiplies $expression$ by $100/\pi$.

This function gives you a way to use a degree angle while in Gradian and Radian mode.

In Degree, Gradian or Radian angle mode:

$cos(45^o) \quad \text{ENTER} \quad \sqrt{2}$

$cos((0^o \cdot (\pi/4^o)^o, 90^o)) \quad \text{ENTER} \quad \{1.707... \cdot 0 \cdot 0.864...\}$

### ã (angle)  $\text{2}^\text{nd} \ [\angle]$ key

<table>
<thead>
<tr>
<th>$radius, \angle$</th>
<th>vector (polar input)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$radius, \angle, \text{Z} _\text{coordinate}$</td>
<td>vector (cylindrical input)</td>
</tr>
<tr>
<td>$radius, \angle, \text{L}_\text{angle}$</td>
<td>vector (spherical input)</td>
</tr>
</tbody>
</table>

Returns coordinates as a vector depending on the vector format selected: rectangular, cylindrical, or spherical.

In Radian mode and vector format set to:

$\{5.60^o, 45^o\} \quad \text{ENTER}$

| $5 \leq 60^o \leq 45^o$ | 5 | 3 | \sqrt{5} |
| $5 \leq 60^o \leq 45^o$ | 3 | 3 | \sqrt{5} |
| $5 \leq 60^o \leq 45^o$ | 5 | \sqrt{5} | 3 | 3 |
### Appendix A: Functions and Instructions

#### (magnitude \( \angle \) angle) \( \Rightarrow \) complexValue (polar input)

Enters a complex value in \((r \angle \theta)\) polar form. The angle is interpreted according to the current Angle mode setting.

<table>
<thead>
<tr>
<th>( \Rightarrow ) complexValue (polar input)</th>
<th>In Radian angle mode and Rectangular complex format mode:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 5 \angle \frac{\pi}{4} )</td>
<td>( -2.071 \ldots -4.071\ldots \angle )</td>
</tr>
</tbody>
</table>

#### \( \text{2nd} \) \( [\ast] \) key, \( \text{2nd} \) \( [-] \) key (‘)

**2nd \( [\ast] \) key (‘), \( \text{2nd} \) \( [-] \) key (‘)**

- \( \text{2nd} \) \( [\ast] \) key (‘): \( \text{2nd} \) \( [\ast] \) key (‘)
- \( \text{2nd} \) \( [-] \) key (‘)

<table>
<thead>
<tr>
<th>dd ( \text{mm} ) ss.ss* ( \Rightarrow ) expression</th>
<th>In Degree angle mode:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( dd ) ( \text{A positive or negative number} )</td>
<td>( 25^\circ 13' 17.5'' )</td>
</tr>
<tr>
<td>( mm ) ( \text{A non-negative number} )</td>
<td>( 25^\circ 30' )</td>
</tr>
<tr>
<td>( ss.ss ) ( \text{A non-negative number} )</td>
<td></td>
</tr>
</tbody>
</table>

Returns \( dd+(mm/60)+(ss.ss /3600) \).

This base-60 entry format lets you:

- Enter an angle in degrees/minutes/seconds without regard to the current angle mode.
- Enter time as hours/minutes/seconds.

#### \( \text{2nd} \) \( [\ast] \) key (\( \prime \))

**2nd \( [\ast] \) key (\( \prime \))**

<table>
<thead>
<tr>
<th>( \text{2nd} ) ( [\ast] ) key (( \prime )) ( \Rightarrow ) expression</th>
<th>deSolve((y''=y^{\prime}(\cdot 1/2)) and ( y(0)=0 ) and ( y'(0)=0),t,y) ( \Rightarrow ) entry</th>
</tr>
</thead>
</table>
| \( variable^* \) \( variable^* \) \( \Rightarrow \) expression \( deSolve(y''=y^{\prime}\cdot 1/2)\) and \( y(0)=0 \) and \( y'(0)=0\),t,y) \( \Rightarrow \) entry | \( \begin{align*}
2y^{\prime\prime} + y &= 0 \\
\frac{2}{3} \cdot y &= t
\end{align*} \) |

#### \( \text{2nd} \) \( [-] \) key (\( _\) underline)

**2nd \( [-] \) key (\( _\) underline)**

<table>
<thead>
<tr>
<th>( \text{expression}_) ( unit ) ( \Rightarrow ) expression</th>
<th>3. ( m) ( \text{ft} )</th>
<th>9.842\ldots ( \text{ft} )</th>
</tr>
</thead>
</table>

**Note:** To type \( \text{^\_} \), press \( 2\text{nd} \) \( [\ast] \).

You can use pre-defined units or create your own units. For a list of pre-defined units, refer to the module about constants and measurement units.

You can press \( 2\text{nd} \) \( \text{Key} \) \( \text{cats} \) to select units from a menu, or you can type the unit names directly.

#### variable_

**variable_**

- When \( variable\) has no value, it is treated as though it represents a complex number. By default, without the \( \_\), the variable is treated as real.
- If \( variable\) has a value, the \( \_\) is ignored and \( variable\) retains its original data type.

**Note:** You can store a complex number to a variable without using \( \_\). However, for best results in calculations such as \( c\text{Solve()} \) and \( c\text{Zeros()} \), the \( \_\) is recommended.

---

910 | Appendix A: Functions and Instructions
Appendix A: Functions and Instructions

### (convert) ▶ key

<table>
<thead>
<tr>
<th>expression_unit</th>
<th>_ _ unit2 ⇒ expression_unit2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3_m ft</td>
<td>9.842…_ft</td>
</tr>
</tbody>
</table>

Converts an expression from one unit to another. The units must be in the same category.

The _ underscore character designates the units. For a list of valid pre-defined units, refer to the module about constants and measurement units. You can press [Units] to select units from a menu, or you can type the unit names directly.

To get the _ underscore when typing units directly, press [¥].

**Note:** The conversion operator does not handle temperature units. Use `tmpCnv()` and `@@ tmpCnv()` instead.

### 10^( )

<table>
<thead>
<tr>
<th><strong>CATALOG</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>10^ (expression) ⇒ expression</td>
</tr>
<tr>
<td>10^ (list) ⇒ list</td>
</tr>
</tbody>
</table>

Returns 10 raised to the power of the argument. For a list, returns 10 raised to the power of the elements in list.

| 10^ (1.5) | 31.622... |
| 10^[0, -2, 2, a] | [1 1 100 10^a] |

### x⁻¹

<table>
<thead>
<tr>
<th><strong>CATALOG</strong> (⁻¹)</th>
</tr>
</thead>
<tbody>
<tr>
<td>expression x⁻¹ ⇒ expression</td>
</tr>
<tr>
<td>list x⁻¹ ⇒ list</td>
</tr>
</tbody>
</table>

Returns the reciprocal of the argument. For a list, returns the reciprocals of the elements in list.

| 3.1⁻¹ | .32581 |
| {a, 4, -1, x²}⁻¹ | {1/a 1/x 1/x²} |

### Square Matrix x⁻¹

<table>
<thead>
<tr>
<th><strong>CATALOG</strong> x⁻¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>squareMatrix x⁻¹ ⇒ squareMatrix</td>
</tr>
</tbody>
</table>

Returns the inverse of squareMatrix. squareMatrix must be a non-singular square matrix.

| [1 2]⁻¹ | [2 1] |
| [3 4]⁻¹ | [3 2]⁻¹ |
| [2 z]⁻¹ | [2 z]⁻¹ |

---

---
The "with" (|) symbol serves as a binary operator. The operand to the left of | is an expression. The operand to the right of | specifies one or more relations that are intended to affect the simplification of the expression. Multiple relations after | must be joined by a logical "and".

The "with" operator provides three basic types of functionality: substitutions, interval constraints, and exclusions.

Substitutions are in the form of an equality, such as \( x = 3 \) or \( y = \sin(x) \). To be most effective, the left side should be a simple variable. \( \text{expression} | \text{variable} = \text{value} \) will substitute value for every occurrence of variable in expression.

Interval constraints take the form of one or more inequalities joined by logical "and" operators. Interval constraints also permit simplification that otherwise might be invalid or not computable.

Exclusions use the "not equals" (!= or ƒ) relational operator to exclude a specific value from consideration. They are used primarily to exclude an exact solution when using \( \text{cSolve()} \), \( \text{cZeros()} \), \( \text{fMax()} \), \( \text{fMin()} \), \( \text{solve()} \), \( \text{zeros()} \), etc.

**Program Editor/Control menu** or \( \text{STOP} \) key

- If variable var does not exist, creates var and initializes it to expression, list, or matrix.
- If var already exists and if it is not locked or protected, replaces its contents with expression, list, or matrix.
- **Hint**: If you plan to do symbolic computations using undefined variables, avoid storing anything into commonly used, one-letter variables such as a, b, c, x, y, z, etc.

**Program segment**:

```
:©:©:©:©:
:©:©:©:©:
:©:©:©:©:
:©:©:©:©:
```

**Appendix A: Functions and Instructions**
### 0b, 0h keys

<table>
<thead>
<tr>
<th>0b binaryNumber</th>
<th>0h hexadecimalNumber</th>
</tr>
</thead>
</table>

Denotes a binary or hexadecimal number, respectively. To enter a binary or hex number, you must enter the 0b or 0h prefix regardless of the Base mode. Without a prefix, a number is treated as decimal (base 10).

Results are displayed according to the Base mode.

In Dec base mode:

```
0b10+0hF+10  \[ENTER\]  27
```

In Bin base mode:

```
0b10+0hF+10  \[ENTER\]  0b11011
```

In Hex base mode:

```
0b10+0hF+10  \[ENTER\]  0h1B
```
This section contains a comprehensive list of TI-89 Titanium / Voyage™ 200 error messages and character codes. It also includes information about how certain TI-89 Titanium / Voyage™ 200 operations are calculated.

**TI-89 Titanium / Voyage™ 200 Error Messages**

This section lists error messages that may be displayed when input or internal errors are encountered. The number to the left of each error message represents an internal error number that is not displayed. If the error occurs inside a Try...EndTry block, the error number is stored in system variable `errornum`. Many of the error messages are self-explanatory and do not require descriptive information. However, additional information has been added for some error messages.

<table>
<thead>
<tr>
<th>Error Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A function did not return a value</td>
</tr>
</tbody>
</table>
| 20           | A test did not resolve to TRUE or FALSE  
  Generally, undefined variables cannot be compared. For example, the test If a<b will cause this error if either a or b is undefined when the If statement is executed. |
| 30           | Argument cannot be a folder name |
| 40           | Argument error |
| 50           | Argument mismatch  
  Two or more arguments must be of the same type. For example, `PtOn expression1,expression2` and `PtOn list1,list2` are both valid, but `PtOn expression,list` is a mismatch. |
| 60           | Argument must be a Boolean expression or integer |
| 70           | Argument must be a decimal number |
| 80           | Argument must be a label name |
| 90           | Argument must be a list |
| 100          | Argument must be a matrix |
| 110          | Argument must be a Pic |
| 120          | Argument must be a Pic or string |
| 130          | Argument must be a string |
| 140          | Argument must be a variable name  
  For example, DelVar 12 is invalid because a number cannot be a variable name. |
| 150          | Argument must be an empty folder name |
| 160          | Argument must be an expression  
  For example, zeros(2x+3=0,x) is invalid because the first argument is an equation. |
<p>| 161          | ASAP or Exec string too long |</p>
<table>
<thead>
<tr>
<th>Error Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>163</td>
<td>Attribute (8-digit number) of object (8-digit number) not found</td>
</tr>
</tbody>
</table>
| 165          | Batteries too low for sending or receiving  
Install new batteries before sending or receiving. |
| 170          | Bound  
For the interactive graph math functions like 2:Zero, the lower bound must be less than the upper bound to define the search interval. |
| 180          | Break  
The key was pressed during a long calculation or during program execution. |
| 185          | Checksum error |
| 190          | Circular definition  
This message is displayed to avoid running out of memory during infinite replacement of variable values during simplification. For example, a+1→a, where a is an undefined variable, will cause this error. |
| 200          | Constraint expression invalid  
For example, solve(3x^2-4=0, x) | x<0 or x>5 would produce this error message because the constraint is separated by “or” and not “and.” |
| 205          | Data is too big to save to a variable. Please use F6 Util to reduce the size.  
The size of the data in the editor exceeds the maximum size that can be saved in a variable. The F6 Util menu provides operations that can be used to reduce the size of the data. |
| 210          | Data type  
An argument is of the wrong data type. |
| 220          | Dependent limit  
A limit of integration is dependent on the integration variable. For example, ∫(x^2,x,1,x) is not allowed. |
| 225          | Diff Eq setup |
| 230          | Dimension  
A list or matrix index is not valid. For example, if the list {1,2,3,4} is stored in L1, then L1[5] is a dimension error because L1 only contains four elements. |
| 240          | Dimension mismatch  
Two or more arguments must be of the same dimension. For example, [1,2]+[1,2,3] is a dimension mismatch because the matrices contain a different number of elements. |
| 250          | Divide by zero |
| 260          | Domain error  
An argument must be in a specified domain. For example, ans(100) is not valid because the argument for ans() must be in the range 1–99. |
<p>| 270          | Duplicate variable name |</p>
<table>
<thead>
<tr>
<th>Error Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>280</td>
<td>Else and ElseIf invalid outside of If..EndIf block</td>
</tr>
<tr>
<td>290</td>
<td>EndTry is missing the matching Else statement</td>
</tr>
<tr>
<td>295</td>
<td>Excessive iteration</td>
</tr>
<tr>
<td>300</td>
<td>Expected 2 or 3-element list or matrix</td>
</tr>
<tr>
<td>307</td>
<td>Flash application extension (function or program) not found</td>
</tr>
<tr>
<td>308</td>
<td>Flash application not found</td>
</tr>
<tr>
<td>310</td>
<td>First argument of nSolve must be a univariate equation</td>
</tr>
<tr>
<td></td>
<td>The first argument must be an equation, and the equation cannot</td>
</tr>
<tr>
<td></td>
<td>contain a non-valued variable other than the variable of interest. For</td>
</tr>
<tr>
<td></td>
<td>example, nSolve(3x^2-4=0, x) is a valid equation; however,</td>
</tr>
<tr>
<td></td>
<td>nSolve(3x^2-4, x) is not an equation, and nSolve(3x^2-y=0,x) is not</td>
</tr>
<tr>
<td></td>
<td>a univariate equation because y has no value in this example.</td>
</tr>
<tr>
<td>320</td>
<td>First argument of solve or cSolve must be an equation or inequality</td>
</tr>
<tr>
<td></td>
<td>For example, solve(3x^2-4, x) is invalid because the first argument is</td>
</tr>
<tr>
<td></td>
<td>not an equation.</td>
</tr>
<tr>
<td>330</td>
<td>Folder</td>
</tr>
<tr>
<td></td>
<td>An attempt was made in the VAR-LINK menu to store a variable in a</td>
</tr>
<tr>
<td></td>
<td>folder that does not exist.</td>
</tr>
<tr>
<td>335</td>
<td>Graph functions y1(x)...y99(x) not available in Diff Equations mode</td>
</tr>
<tr>
<td>345</td>
<td>Inconsistent units</td>
</tr>
<tr>
<td>350</td>
<td>Index out of range</td>
</tr>
<tr>
<td>360</td>
<td>Indirection string is not a valid variable name</td>
</tr>
<tr>
<td>380</td>
<td>Invalid ans()</td>
</tr>
<tr>
<td>390</td>
<td>Invalid assignment</td>
</tr>
<tr>
<td>400</td>
<td>Invalid assignment value</td>
</tr>
<tr>
<td>405</td>
<td>Invalid axes</td>
</tr>
<tr>
<td>410</td>
<td>Invalid command</td>
</tr>
<tr>
<td>420</td>
<td>Invalid folder name</td>
</tr>
<tr>
<td>430</td>
<td>Invalid for the current mode settings</td>
</tr>
<tr>
<td>440</td>
<td>Invalid implied multiply</td>
</tr>
<tr>
<td></td>
<td>For example, x(x+1) is invalid; whereas, x*(x+1) is the correct</td>
</tr>
<tr>
<td></td>
<td>syntax. This is to avoid confusion between implied multiplication and</td>
</tr>
<tr>
<td></td>
<td>function calls.</td>
</tr>
<tr>
<td>450</td>
<td>Invalid in a function or current expression</td>
</tr>
<tr>
<td></td>
<td>Only certain commands are valid in a user-defined function. Entries</td>
</tr>
<tr>
<td></td>
<td>that are made in the Window Editor, Table Editor, Data/Matrix Editor,</td>
</tr>
<tr>
<td></td>
<td>and Solver as well as system prompts such as Lower Bound cannot</td>
</tr>
<tr>
<td></td>
<td>contain any commands or a colon (:). See also “Creating and</td>
</tr>
<tr>
<td></td>
<td>Evaluating User-Defined Functions” in the Calculator Home Screen</td>
</tr>
<tr>
<td></td>
<td>module.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>460</td>
<td>Invalid in Custom..EndCustm block</td>
</tr>
<tr>
<td>470</td>
<td>Invalid in Dialog..EndDlog block</td>
</tr>
<tr>
<td>480</td>
<td>Invalid in Toolbar..EndTBar block</td>
</tr>
<tr>
<td>490</td>
<td>Invalid in Try..EndTry block</td>
</tr>
<tr>
<td>500</td>
<td>Invalid label</td>
</tr>
<tr>
<td></td>
<td>Label names must follow the same rules used for naming variables.</td>
</tr>
<tr>
<td>510</td>
<td>Invalid list or matrix</td>
</tr>
<tr>
<td></td>
<td>For example, a list inside a list such as (2,(3,4)) is not valid.</td>
</tr>
<tr>
<td>520</td>
<td>Invalid outside Custom..EndCustm or Toolbar..EndTBar blocks</td>
</tr>
<tr>
<td></td>
<td>For example, an Item command is attempted outside a Custom or Toolbar structure.</td>
</tr>
<tr>
<td>530</td>
<td>Invalid outside Dialog..EndDlog, Custom..EndCustm, or Toolbar..EndTBar blocks</td>
</tr>
<tr>
<td></td>
<td>For example, a Title command is attempted outside a Dialog, Custom, or Toolbar structure.</td>
</tr>
<tr>
<td>540</td>
<td>Invalid outside Dialog..EndDlog block</td>
</tr>
<tr>
<td></td>
<td>For example, the DropDownList command is attempted outside a Dialog structure.</td>
</tr>
<tr>
<td>550</td>
<td>Invalid outside function or program</td>
</tr>
<tr>
<td></td>
<td>A number of commands are not valid outside a program or a function. For example, Local cannot be used unless it is in a program or function.</td>
</tr>
<tr>
<td>560</td>
<td>Invalid outside Loop..EndLoop, For..EndFor, or While..EndWhile blocks</td>
</tr>
<tr>
<td></td>
<td>For example, the Exit command is valid only inside these loop blocks.</td>
</tr>
<tr>
<td>570</td>
<td>Invalid pathname</td>
</tr>
<tr>
<td></td>
<td>For example, \var is invalid.</td>
</tr>
<tr>
<td>575</td>
<td>Invalid polar complex</td>
</tr>
<tr>
<td>580</td>
<td>Invalid program reference</td>
</tr>
<tr>
<td></td>
<td>Programs cannot be referenced within functions or expressions such as 1+p(x) where p is a program.</td>
</tr>
<tr>
<td>585</td>
<td>Invalid relocation data in ASM program</td>
</tr>
<tr>
<td></td>
<td>The necessary relocation data in the ASM (Assembly) program is missing or corrupted.</td>
</tr>
<tr>
<td>590</td>
<td>Invalid syntax block</td>
</tr>
<tr>
<td></td>
<td>A Dialog..EndDlog block is empty or has more than one title. A Custom..EndCustm block cannot contain PIC variables, and items must be preceded by a title. A Toolbar..EndTBar block must have a second argument if no items follow; or items must have a second argument and must be preceded by a title.</td>
</tr>
<tr>
<td>600</td>
<td>Invalid table</td>
</tr>
<tr>
<td>605</td>
<td>Invalid use of units</td>
</tr>
<tr>
<td>Error Number</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>610</td>
<td>Invalid variable name in a Local statement</td>
</tr>
<tr>
<td>620</td>
<td>Invalid variable or function name</td>
</tr>
<tr>
<td>630</td>
<td>Invalid variable reference</td>
</tr>
<tr>
<td>640</td>
<td>Invalid vector syntax</td>
</tr>
</tbody>
</table>
| 650          | Link transmission  
A transmission between two units was not completed. Verify that the connecting cable is connected firmly to both units. |
| 665          | Matrix not diagonalizable |
| 670          | Memory  
The calculation required more memory than was available at that time. If you get this error when you run a large program, you may need to break the program into separate, smaller programs or functions (where one program or function calls another). |
| 680          | Missing ( |
| 690          | Missing ) |
| 700          | Missing " |
| 710          | Missing ] |
| 720          | Missing ) |
| 730          | Missing start or end of block syntax |
| 740          | Missing Then in the If..EndIf block |
| 750          | Name is not a function or program |
| 765          | No functions selected |
| 780          | No solution found  
Using the interactive math features (F5:Math) in the Graph application can give this error. For example, if you attempt to find an inflection point of the parabola $y_1(x)=x^2$, which does not exist, this error will be displayed. |
| 790          | Non-algebraic variable in expression  
If a is the name of a PIC, GDB, MAC, FIG, etc., a+1 is invalid. Use a different variable name in the expression or delete the variable. |
| 800          | Non-real result  
For example, if the unit is in the REAL setting of the Complex Format mode, $\ln(-2)$ is invalid. |
| 810          | Not enough memory to save current variable. Please delete unneeded variables on the Var-Link screen and re-open editor as current OR re-open editor and use F1 8 to clear editor.  
This error message is caused by very low memory conditions inside the Data/Matrix Editor. |
<p>| 830          | Overflow |
| 840          | Plot setup |</p>
<table>
<thead>
<tr>
<th>Error Number</th>
<th>Description</th>
</tr>
</thead>
</table>
| 850          | Program not found  
   A program reference inside another program could not be found in the provided path during execution. |
| 855          | Rand type functions not allowed in 3D graphing |
| 860          | Recursion is limited to 255 calls deep |
| 870          | Reserved name or system variable |
| 875          | ROM-resident routine not available |
| 880          | Sequence setup |
| 885          | Signature error |
| 890          | Singular matrix |
| 895          | Slope fields need one selected function and are used for 1st-order equations only |
| 900          | Stat |
| 910          | Syntax  
   The structure of the entry is incorrect. For example, x+ - y (x plus minus y) is invalid; whereas, x+-y (x plus negative y) is correct. |
| 930          | Too few arguments  
   The expression or equation is missing one or more arguments. For example, d(f(x)) is invalid; whereas, d(f(x),x) is the correct syntax. |
| 940          | Too many arguments  
   The expression or equation contains an excessive number of arguments and cannot be evaluated. |
| 950          | Too many subscripts |
| 955          | Too many undefined variables |
| 960          | Undefined variable |
| 965          | Unlicensed OS or Flash application |
| 970          | Variable in use so references or changes are not allowed |
| 980          | Variable is locked, protected, or archived |
| 990          | Variable name is limited to 8 characters |
| 1000         | Window variables domain |
| 1010         | Zoom  
   Warning: \( \infty^0 \) or \( \text{undef}^0 \) replaced by 1  
   Warning: \( 0^0 \) replaced by 1  
   Warning: \( 1^{\infty} \) or \( 1^{\text{undef}} \) replaced by 1  
   Warning: cSolve may specify more zeros  
   Warning: May produce false equation |
This section describes the modes of the TI-89 Titanium / Voyage™ 200 and lists the possible settings of each mode. These mode settings are displayed when you press 3.

### Graph

Specifies the type of graphs you can plot.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:FUNCTION</td>
<td>y(x) functions</td>
</tr>
<tr>
<td>2:PARAMETRIC</td>
<td>x(t) and y(t) parametric equations</td>
</tr>
<tr>
<td>3:POLAR</td>
<td>r(θ) polar equations</td>
</tr>
<tr>
<td>4:SEQUENCE</td>
<td>u(n) sequences</td>
</tr>
<tr>
<td>5:3D</td>
<td>z(x,y) 3D equations</td>
</tr>
<tr>
<td>6:DIFF EQUATIONS</td>
<td>y’(t) differential equations</td>
</tr>
</tbody>
</table>

**Note:** If you use a split screen with Number of Graphs = 2, Graph is for the top or left part of the screen and Graph 2 is for the bottom or right part.

### Current Folder

Specifies the current folder. You can set up multiple folders with unique configurations of variables, graph databases, programs, etc.

---

**TI-89 Titanium / Voyage™ 200 Modes**

Warning: Expected finite real integrand
Warning: May not be fully simplified
Warning: More solutions may exist
Warning: May introduce false solutions
Warning: Operation may lose solutions
Warning: Requires & returns 32 bit value
Warning: Overflow replaced by $\infty$ or $-\infty$
Warning: Questionable accuracy
Warning: Questionable solution
Warning: Solve may specify more zeros
Warning: Trig argument too big to reduce
Warning: Non-real intermediate result
Note: Domain of result may be larger
Note: Domain of result may be smaller
Note: For detailed information about using folders, see *Calculator Home Screen*.

| 1:main | Default folder included with the TI-89 Titanium / Voyage™ 200. |
| 2: — | Other folders are available only if they have been created by a user. |

### Display Digits

Selects the number of digits. These decimal settings affect only how results are displayed—you can enter a number in any format.

Internally, the TI-89 Titanium / Voyage™ 200 retains decimal numbers with 14 significant digits. For display purposes, such numbers are rounded to a maximum of 12 significant digits.

| 1:FIX 0 | Results are always displayed with the selected number of decimal places. |
| 2:FIX 1 | |
| ... | |
| D:FIX 12 | |
| E:FLOAT | The number of decimal places varies, depending on the result. |
| F:FLOAT 1 | If the integer part has more than the selected number of digits, the result is rounded and displayed in scientific notation. |
| G:FLOAT 2 | ... |
| Q:FLOAT 12 | For example, in FLOAT 4: 12345. is shown as 1.235E4 |

### Angle

Specifies the units in which angle values are interpreted and displayed in trig functions and polar/rectangular conversions.

| 1:RADIAN | |
| 2:DEGREE | |
| 3:GRADIAN | |

### Exponential Format

Specifies which notation format should be used. These formats affect only how an answer is displayed; you can enter a number in any format. Numeric answers can be displayed with up to 12 digits and a 3-digit exponent.

| 1:NORMAL | Expresses numbers in standard format. For example, 12345.67 |
| 2:SCIENTIFIC | Expresses numbers in two parts:  
  - The significant digits display with one digit to the left of the decimal.  
  - The power of 10 displays to the right of \( \times \).  
  For example, 1.234567\( \times \)10^4 |
Note: If you select NORMAL, but the answer cannot be displayed in the number of digits selected by Display Digits, the TI-89 Titanium / Voyage™ 200 displays the answer in SCIENTIFIC notation. If Display Digits = FLOAT, scientific notation will be used for exponents of 12 or more and exponents of -4 or less.

### Complex Format

Specifies whether complex results are displayed and, if so, their format.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:REAL</td>
<td>Does not display complex results. (If a result is a complex number and the input does not contain the complex unit $i$, an error message is displayed.)</td>
</tr>
<tr>
<td>2:RECTANGULAR</td>
<td>Displays complex numbers in the form: $a+bi$</td>
</tr>
<tr>
<td>3:POLAR</td>
<td>Displays complex numbers in the form: $re^{i\theta}$</td>
</tr>
</tbody>
</table>

### Vector Format

Determines how 2-element and 3-element vectors are displayed. You can enter vectors in any of the coordinate systems.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:RECTANGULAR</td>
<td>Coordinates are in terms of $x$, $y$, and $z$. For example, $[3,5,2]$ represents $x = 3$, $y = 5$, and $z = 2$.</td>
</tr>
<tr>
<td>2:CYLINDRICAL</td>
<td>Coordinates are in terms of $r$, $\theta$, and $z$. For example, $[3,\angle45,2]$ represents $r = 3$, $\theta = 45$, and $z = 2$.</td>
</tr>
<tr>
<td>3:SPHERICAL</td>
<td>Coordinates are in terms of $r$, $\theta$, and $\phi$. For example, $[3,\angle45,\angle90]$ represents $r = 3$, $\theta = 45$, and $\phi = 90$.</td>
</tr>
</tbody>
</table>

### Pretty Print

Determines how results are displayed on the Home screen.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:OFF</td>
<td>Results are displayed in a linear, one-dimensional form. For example, $\pi^2$, $\pi/2$, or $\sqrt{(x-3)/x}$</td>
</tr>
<tr>
<td>2:ON</td>
<td>Results are displayed in conventional mathematical format. For example, $\pi^2$, $\frac{\pi}{2}$, or $\sqrt{x - \frac{3}{x}}$</td>
</tr>
</tbody>
</table>

Note: For a complete description of these settings, refer to “Formats of Displayed Results” in the *Operating the Calculator* module.
Split Screen

Lets you split the screen into two parts. For example, you can display a graph and see the Y= Editor at the same time.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:FULL</td>
<td>The screen is not split.</td>
</tr>
<tr>
<td>2:TOP-BOTTOM</td>
<td>The applications are shown in two screens that are above and below each other.</td>
</tr>
<tr>
<td>3:LEFT-RIGHT</td>
<td>The applications are shown in two screens that are to the left and right of each other.</td>
</tr>
</tbody>
</table>

To determine what and how information is displayed on a split screen, use this mode in conjunction with other modes such as Split 1 App, Split 2 App, Number of Graphs, and Split Screen Ratio. (Split Screen Ratio is available on the Voyage™ 200 only.)

**Split 1 App and Split 2 App**

Specifies which application is displayed on the screen.

- For a full screen, only Split 1 App is active.
- For a split screen, Split 1 App is the top or left part of the screen and Split 2 App is the bottom or right part.

The available application choices are those listed when you press \( \text{B} \) from the Page 2 mode screen or when you press \( \text{O} \). You must have different applications in each screen unless you are in 2-graph mode.

**Number of Graphs**

Specifies whether both parts of a split screen can display graphs at the same time.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Only one part can display graphs.</td>
</tr>
<tr>
<td>2</td>
<td>Both parts can display an independent graph screen (Graph or Graph 2 setting) with independent settings.</td>
</tr>
</tbody>
</table>

**Graph 2**

Specifies the type of graphs that you can plot for the second graph on a two-graph split screen. This is active only when Number of Graphs = 2. In this two-graph setting, Graph sets the type of graph for the top or left part of the split screen, and Graph 2 sets the bottom or right part. The available choices are the same as for Graph.

**Split Screen Ratio (Voyage 200 only)**

Specifies the proportional sizes of the two parts of a split screen.

<table>
<thead>
<tr>
<th>Ratio</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:1</td>
<td>The screen is split evenly.</td>
</tr>
<tr>
<td>1:2</td>
<td>The bottom or right part is approximately twice the size of the top or left part.</td>
</tr>
<tr>
<td>2:1</td>
<td>The top or left part is approximately twice the size of the bottom or right part.</td>
</tr>
</tbody>
</table>
## Exact/Approx

Specifies how fractional and symbolic expressions are calculated and displayed. By retaining rational and symbolic forms in the EXACT setting, the TI-89 Titanium / Voyage™ 200 increases precision by eliminating most numeric rounding errors.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:AUTO</td>
<td>Uses EXACT setting in most cases. However, uses APPROXIMATE if the entry contains a decimal point.</td>
</tr>
<tr>
<td>2:EXACT</td>
<td>Displays non-whole-number results in their rational or symbolic form.</td>
</tr>
<tr>
<td>3:APPROXIMATE</td>
<td>Displays numeric results in floating-point form.</td>
</tr>
</tbody>
</table>

**Note:** For a complete description of these settings, refer to “Formats of Displayed Results” in the Operating the Calculator module.

## Base

Lets you perform calculations by entering numbers in decimal, binary, or hexadecimal form.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:DEC</td>
<td>Decimal numbers use 0 - 9 in the base 10 format</td>
</tr>
<tr>
<td>2:HEX</td>
<td>Hexadecimal numbers use 0 - 9 and A - F in the base 16 format.</td>
</tr>
<tr>
<td>3:BIN</td>
<td>Binary numbers use 0 and 1 in the base 2 format.</td>
</tr>
</tbody>
</table>
### Unit System

Lets you enter a unit for values in an expression, such as $6\_m \times 4\_m$ or $23\_/s \times 10\_s$, convert values from one unit to another within the same category, and create your own user-defined units.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:SI</td>
<td>Select SI for the metric system of measurements</td>
</tr>
<tr>
<td>2:ENG/US</td>
<td>Select ENG/US for the non-metric system of measurements</td>
</tr>
<tr>
<td>3:CUSTOM</td>
<td>Allows you to select custom defaults.</td>
</tr>
</tbody>
</table>

### Custom Units

Lets you select custom defaults. This mode is dimmed until you select Unit System, 3:CUSTOM.

### Language

Lets you localize the TI-89 Titanium / Voyage™ 200 into one of several languages, depending on which language Flash applications are installed.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:English</td>
<td>Default language included with the TI-89 Titanium / Voyage™ 200 operating system (OS).</td>
</tr>
<tr>
<td>2: — (language Flash applications)</td>
<td>Alternate languages are available only if the respective language Flash applications have been installed.</td>
</tr>
</tbody>
</table>

### Apps Desktop

Lets you turn the display of the Apps desktop on or off.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>Displays the navigable Apps desktop. The Apps desktop appears when you:</td>
</tr>
<tr>
<td></td>
<td>• Press [APPS].</td>
</tr>
<tr>
<td></td>
<td>• Turn the unit on after it has been turned off by pressing [2nd] [OFF].</td>
</tr>
<tr>
<td></td>
<td>• Press [2nd] [QUIT] from an App that is displayed in full screen mode.</td>
</tr>
<tr>
<td>OFF</td>
<td>• Does not display the navigable Apps desktop.</td>
</tr>
<tr>
<td></td>
<td>• The unit defaults to the calculator Home screen.</td>
</tr>
<tr>
<td></td>
<td>• The calculator Home screen displays when you press [2nd] [QUIT].</td>
</tr>
<tr>
<td></td>
<td>• The APPLICATIONS menu displays when you press [APPS].</td>
</tr>
</tbody>
</table>
The `char()` function lets you refer to any character by its numeric character code. For example, to display `2` on the Program I/O screen, use `Disp char(127)`. You can use `ord()` to find the numeric code of a character. For example, `ord("A")` returns 65.

<table>
<thead>
<tr>
<th>Code</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOH</td>
</tr>
<tr>
<td>2</td>
<td>STX</td>
</tr>
<tr>
<td>3</td>
<td>ETX</td>
</tr>
<tr>
<td>4</td>
<td>EOT</td>
</tr>
<tr>
<td>5</td>
<td>ENQ</td>
</tr>
<tr>
<td>6</td>
<td>ACK</td>
</tr>
<tr>
<td>7</td>
<td>BS</td>
</tr>
<tr>
<td>8</td>
<td>LS</td>
</tr>
<tr>
<td>9</td>
<td>TAB</td>
</tr>
<tr>
<td>10</td>
<td>LF</td>
</tr>
<tr>
<td>11</td>
<td>DH</td>
</tr>
<tr>
<td>12</td>
<td>FF</td>
</tr>
<tr>
<td>13</td>
<td>CR</td>
</tr>
<tr>
<td>14</td>
<td>SUB</td>
</tr>
<tr>
<td>15</td>
<td>CTRL</td>
</tr>
<tr>
<td>16</td>
<td>&quot;</td>
</tr>
<tr>
<td>17</td>
<td>#</td>
</tr>
<tr>
<td>18</td>
<td>$</td>
</tr>
<tr>
<td>19</td>
<td>%</td>
</tr>
<tr>
<td>20</td>
<td>^</td>
</tr>
<tr>
<td>21</td>
<td>_</td>
</tr>
<tr>
<td>22</td>
<td>`</td>
</tr>
<tr>
<td>23</td>
<td>a</td>
</tr>
<tr>
<td>24</td>
<td>b</td>
</tr>
<tr>
<td>25</td>
<td>c</td>
</tr>
<tr>
<td>26</td>
<td>d</td>
</tr>
<tr>
<td>27</td>
<td>e</td>
</tr>
<tr>
<td>28</td>
<td>f</td>
</tr>
<tr>
<td>29</td>
<td>g</td>
</tr>
<tr>
<td>30</td>
<td>h</td>
</tr>
<tr>
<td>31</td>
<td>i</td>
</tr>
<tr>
<td>32</td>
<td>SPACE</td>
</tr>
<tr>
<td>33</td>
<td>!</td>
</tr>
<tr>
<td>34</td>
<td>&quot;</td>
</tr>
<tr>
<td>35</td>
<td>#</td>
</tr>
<tr>
<td>36</td>
<td>$</td>
</tr>
<tr>
<td>37</td>
<td>%</td>
</tr>
<tr>
<td>38</td>
<td>&amp;</td>
</tr>
<tr>
<td>39</td>
<td>'</td>
</tr>
<tr>
<td>40</td>
<td>(</td>
</tr>
<tr>
<td>41</td>
<td>)</td>
</tr>
<tr>
<td>42</td>
<td>*</td>
</tr>
<tr>
<td>43</td>
<td>+</td>
</tr>
<tr>
<td>44</td>
<td>-</td>
</tr>
<tr>
<td>45</td>
<td>.</td>
</tr>
<tr>
<td>46</td>
<td>/</td>
</tr>
<tr>
<td>47</td>
<td>0</td>
</tr>
<tr>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>49</td>
<td>2</td>
</tr>
<tr>
<td>50</td>
<td>3</td>
</tr>
<tr>
<td>51</td>
<td>4</td>
</tr>
<tr>
<td>52</td>
<td>5</td>
</tr>
<tr>
<td>53</td>
<td>6</td>
</tr>
<tr>
<td>54</td>
<td>7</td>
</tr>
<tr>
<td>55</td>
<td>8</td>
</tr>
<tr>
<td>56</td>
<td>9</td>
</tr>
<tr>
<td>57</td>
<td>:</td>
</tr>
<tr>
<td>58</td>
<td>;</td>
</tr>
<tr>
<td>59</td>
<td>=</td>
</tr>
<tr>
<td>60</td>
<td>&gt;</td>
</tr>
<tr>
<td>61</td>
<td>?</td>
</tr>
<tr>
<td>62</td>
<td>&lt;</td>
</tr>
<tr>
<td>63</td>
<td>&gt;</td>
</tr>
<tr>
<td>64</td>
<td>@</td>
</tr>
<tr>
<td>65</td>
<td>A</td>
</tr>
<tr>
<td>66</td>
<td>B</td>
</tr>
<tr>
<td>67</td>
<td>C</td>
</tr>
<tr>
<td>68</td>
<td>D</td>
</tr>
<tr>
<td>69</td>
<td>E</td>
</tr>
<tr>
<td>70</td>
<td>F</td>
</tr>
<tr>
<td>71</td>
<td>G</td>
</tr>
<tr>
<td>72</td>
<td>H</td>
</tr>
<tr>
<td>73</td>
<td>I</td>
</tr>
<tr>
<td>74</td>
<td>J</td>
</tr>
<tr>
<td>75</td>
<td>K</td>
</tr>
<tr>
<td>76</td>
<td>L</td>
</tr>
<tr>
<td>77</td>
<td>M</td>
</tr>
<tr>
<td>78</td>
<td>N</td>
</tr>
<tr>
<td>79</td>
<td>O</td>
</tr>
<tr>
<td>80</td>
<td>P</td>
</tr>
<tr>
<td>81</td>
<td>Q</td>
</tr>
<tr>
<td>82</td>
<td>R</td>
</tr>
<tr>
<td>83</td>
<td>S</td>
</tr>
<tr>
<td>84</td>
<td>T</td>
</tr>
<tr>
<td>85</td>
<td>U</td>
</tr>
<tr>
<td>86</td>
<td>V</td>
</tr>
<tr>
<td>87</td>
<td>W</td>
</tr>
<tr>
<td>88</td>
<td>X</td>
</tr>
<tr>
<td>89</td>
<td>Y</td>
</tr>
<tr>
<td>90</td>
<td>Z</td>
</tr>
<tr>
<td>91</td>
<td>[</td>
</tr>
<tr>
<td>92</td>
<td>]</td>
</tr>
<tr>
<td>93</td>
<td>]</td>
</tr>
<tr>
<td>94</td>
<td>^</td>
</tr>
<tr>
<td>95</td>
<td>_</td>
</tr>
<tr>
<td>96</td>
<td>`</td>
</tr>
<tr>
<td>97</td>
<td>a</td>
</tr>
<tr>
<td>98</td>
<td>b</td>
</tr>
<tr>
<td>99</td>
<td>c</td>
</tr>
<tr>
<td>100</td>
<td>d</td>
</tr>
<tr>
<td>101</td>
<td>e</td>
</tr>
<tr>
<td>102</td>
<td>f</td>
</tr>
<tr>
<td>103</td>
<td>g</td>
</tr>
<tr>
<td>104</td>
<td>h</td>
</tr>
<tr>
<td>105</td>
<td>i</td>
</tr>
<tr>
<td>106</td>
<td>j</td>
</tr>
<tr>
<td>107</td>
<td>k</td>
</tr>
<tr>
<td>108</td>
<td>l</td>
</tr>
<tr>
<td>109</td>
<td>m</td>
</tr>
<tr>
<td>110</td>
<td>n</td>
</tr>
<tr>
<td>111</td>
<td>o</td>
</tr>
<tr>
<td>112</td>
<td>p</td>
</tr>
<tr>
<td>113</td>
<td>q</td>
</tr>
<tr>
<td>114</td>
<td>r</td>
</tr>
<tr>
<td>115</td>
<td>s</td>
</tr>
<tr>
<td>116</td>
<td>t</td>
</tr>
<tr>
<td>117</td>
<td>u</td>
</tr>
<tr>
<td>118</td>
<td>v</td>
</tr>
<tr>
<td>119</td>
<td>w</td>
</tr>
<tr>
<td>120</td>
<td>x</td>
</tr>
<tr>
<td>121</td>
<td>y</td>
</tr>
<tr>
<td>122</td>
<td>z</td>
</tr>
<tr>
<td>123</td>
<td>{</td>
</tr>
<tr>
<td>124</td>
<td>}</td>
</tr>
<tr>
<td>125</td>
<td>}</td>
</tr>
<tr>
<td>126</td>
<td>~</td>
</tr>
<tr>
<td>127</td>
<td>°</td>
</tr>
<tr>
<td>128</td>
<td>α</td>
</tr>
<tr>
<td>129</td>
<td>β</td>
</tr>
<tr>
<td>130</td>
<td>Γ</td>
</tr>
<tr>
<td>131</td>
<td>γ</td>
</tr>
<tr>
<td>132</td>
<td>δ</td>
</tr>
<tr>
<td>133</td>
<td>ε</td>
</tr>
<tr>
<td>134</td>
<td>ζ</td>
</tr>
<tr>
<td>135</td>
<td>η</td>
</tr>
<tr>
<td>136</td>
<td>θ</td>
</tr>
<tr>
<td>137</td>
<td>ι</td>
</tr>
<tr>
<td>138</td>
<td>ξ</td>
</tr>
<tr>
<td>139</td>
<td>ο</td>
</tr>
<tr>
<td>140</td>
<td>π</td>
</tr>
<tr>
<td>141</td>
<td>ρ</td>
</tr>
<tr>
<td>142</td>
<td>σ</td>
</tr>
<tr>
<td>143</td>
<td>τ</td>
</tr>
<tr>
<td>144</td>
<td>υ</td>
</tr>
<tr>
<td>145</td>
<td>ψ</td>
</tr>
<tr>
<td>146</td>
<td>φ</td>
</tr>
<tr>
<td>147</td>
<td>Ω</td>
</tr>
<tr>
<td>148</td>
<td>ω</td>
</tr>
<tr>
<td>149</td>
<td>Ε</td>
</tr>
<tr>
<td>150</td>
<td>Φ</td>
</tr>
<tr>
<td>151</td>
<td>ι</td>
</tr>
<tr>
<td>152</td>
<td>ζ</td>
</tr>
<tr>
<td>153</td>
<td>ξ</td>
</tr>
<tr>
<td>154</td>
<td>η</td>
</tr>
<tr>
<td>155</td>
<td>θ</td>
</tr>
<tr>
<td>156</td>
<td>ι</td>
</tr>
<tr>
<td>157</td>
<td>η</td>
</tr>
<tr>
<td>158</td>
<td>ζ</td>
</tr>
<tr>
<td>159</td>
<td>η</td>
</tr>
<tr>
<td>160</td>
<td>ι</td>
</tr>
<tr>
<td>161</td>
<td>ι</td>
</tr>
<tr>
<td>162</td>
<td>ι</td>
</tr>
<tr>
<td>163</td>
<td>ι</td>
</tr>
<tr>
<td>164</td>
<td>ι</td>
</tr>
<tr>
<td>165</td>
<td>ι</td>
</tr>
<tr>
<td>166</td>
<td>ι</td>
</tr>
<tr>
<td>167</td>
<td>ι</td>
</tr>
<tr>
<td>168</td>
<td>ι</td>
</tr>
<tr>
<td>169</td>
<td>ι</td>
</tr>
<tr>
<td>170</td>
<td>ι</td>
</tr>
<tr>
<td>171</td>
<td>ι</td>
</tr>
<tr>
<td>172</td>
<td>ι</td>
</tr>
<tr>
<td>173</td>
<td>ι</td>
</tr>
<tr>
<td>174</td>
<td>ι</td>
</tr>
<tr>
<td>175</td>
<td>ι</td>
</tr>
<tr>
<td>176</td>
<td>ι</td>
</tr>
<tr>
<td>177</td>
<td>ι</td>
</tr>
<tr>
<td>178</td>
<td>ι</td>
</tr>
<tr>
<td>179</td>
<td>ι</td>
</tr>
<tr>
<td>180</td>
<td>ι</td>
</tr>
<tr>
<td>181</td>
<td>ι</td>
</tr>
<tr>
<td>182</td>
<td>ι</td>
</tr>
<tr>
<td>183</td>
<td>ι</td>
</tr>
<tr>
<td>184</td>
<td>ι</td>
</tr>
<tr>
<td>185</td>
<td>ι</td>
</tr>
<tr>
<td>186</td>
<td>ι</td>
</tr>
<tr>
<td>187</td>
<td>ι</td>
</tr>
<tr>
<td>188</td>
<td>ι</td>
</tr>
<tr>
<td>189</td>
<td>ι</td>
</tr>
<tr>
<td>190</td>
<td>ι</td>
</tr>
<tr>
<td>191</td>
<td>ι</td>
</tr>
<tr>
<td>192</td>
<td>ι</td>
</tr>
<tr>
<td>193</td>
<td>ι</td>
</tr>
<tr>
<td>194</td>
<td>ι</td>
</tr>
<tr>
<td>195</td>
<td>ι</td>
</tr>
<tr>
<td>196</td>
<td>ι</td>
</tr>
<tr>
<td>197</td>
<td>ι</td>
</tr>
<tr>
<td>198</td>
<td>ι</td>
</tr>
<tr>
<td>199</td>
<td>ι</td>
</tr>
<tr>
<td>200</td>
<td>ι</td>
</tr>
<tr>
<td>201</td>
<td>ι</td>
</tr>
<tr>
<td>202</td>
<td>ι</td>
</tr>
<tr>
<td>203</td>
<td>ι</td>
</tr>
<tr>
<td>204</td>
<td>ι</td>
</tr>
<tr>
<td>205</td>
<td>ι</td>
</tr>
<tr>
<td>206</td>
<td>ι</td>
</tr>
<tr>
<td>207</td>
<td>ι</td>
</tr>
<tr>
<td>208</td>
<td>ι</td>
</tr>
<tr>
<td>209</td>
<td>ι</td>
</tr>
<tr>
<td>210</td>
<td>ι</td>
</tr>
<tr>
<td>211</td>
<td>ι</td>
</tr>
<tr>
<td>212</td>
<td>ι</td>
</tr>
<tr>
<td>213</td>
<td>ι</td>
</tr>
<tr>
<td>214</td>
<td>ι</td>
</tr>
<tr>
<td>215</td>
<td>ι</td>
</tr>
<tr>
<td>216</td>
<td>ι</td>
</tr>
<tr>
<td>217</td>
<td>ι</td>
</tr>
<tr>
<td>218</td>
<td>ι</td>
</tr>
<tr>
<td>219</td>
<td>ι</td>
</tr>
<tr>
<td>220</td>
<td>ι</td>
</tr>
<tr>
<td>221</td>
<td>ι</td>
</tr>
<tr>
<td>222</td>
<td>ι</td>
</tr>
</tbody>
</table>
The `getKey()` function returns a value that corresponds to the last key pressed, according to the tables shown in this section. For example, if your program contains a `getKey()` function, pressing `2nd [F6]` will return a value of 273.

**Table 1: Key Codes for Primary Keys**

<table>
<thead>
<tr>
<th>Key</th>
<th>Modifier</th>
<th>None</th>
<th>2nd</th>
<th>α</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>F1</td>
<td>268</td>
<td>F1</td>
<td>268</td>
<td>F1</td>
</tr>
<tr>
<td>F2</td>
<td>F2</td>
<td>269</td>
<td>F2</td>
<td>269</td>
<td>F2</td>
</tr>
<tr>
<td>F3</td>
<td>F3</td>
<td>270</td>
<td>F3</td>
<td>270</td>
<td>F3</td>
</tr>
<tr>
<td>F4</td>
<td>F4</td>
<td>271</td>
<td>F4</td>
<td>271</td>
<td>F4</td>
</tr>
<tr>
<td>F5</td>
<td>F5</td>
<td>272</td>
<td>F5</td>
<td>272</td>
<td>F5</td>
</tr>
<tr>
<td>a-lock</td>
<td>24576</td>
<td>Cut</td>
<td>12288</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESC</td>
<td>ESC</td>
<td>264</td>
<td>ESC</td>
<td>264</td>
<td>QUIT</td>
</tr>
<tr>
<td>APPS</td>
<td>APPS</td>
<td>265</td>
<td>APPS</td>
<td>265</td>
<td>Switch</td>
</tr>
<tr>
<td>HOME</td>
<td>HOME</td>
<td>277</td>
<td>HOME</td>
<td>277</td>
<td>CUST</td>
</tr>
<tr>
<td>MODE</td>
<td>MODE</td>
<td>266</td>
<td>MODE</td>
<td>266</td>
<td>18</td>
</tr>
<tr>
<td>CATALOG</td>
<td>CATALOG</td>
<td>278</td>
<td>CATALOG</td>
<td>278</td>
<td>i</td>
</tr>
<tr>
<td>BS</td>
<td>BS</td>
<td>257</td>
<td>BS</td>
<td>257</td>
<td>INS</td>
</tr>
<tr>
<td>CLEAR</td>
<td>CLEAR</td>
<td>263</td>
<td>CLEAR</td>
<td>263</td>
<td>CLEAR</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td>120</td>
<td>X</td>
<td>88</td>
<td>LN</td>
</tr>
<tr>
<td>Y</td>
<td>y</td>
<td>121</td>
<td>Y</td>
<td>89</td>
<td>SIN</td>
</tr>
<tr>
<td>Z</td>
<td>z</td>
<td>122</td>
<td>Z</td>
<td>90</td>
<td>COS</td>
</tr>
<tr>
<td>T</td>
<td>t</td>
<td>116</td>
<td>T</td>
<td>84</td>
<td>TAN</td>
</tr>
<tr>
<td>^</td>
<td>94</td>
<td>^</td>
<td>94</td>
<td>π</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td>124</td>
<td>F</td>
<td>70</td>
<td>°</td>
<td>176</td>
</tr>
<tr>
<td>(</td>
<td>40</td>
<td>B</td>
<td>66</td>
<td>}</td>
<td>123</td>
</tr>
<tr>
<td>)</td>
<td>41</td>
<td>C</td>
<td>67</td>
<td>}</td>
<td>125</td>
</tr>
<tr>
<td>,</td>
<td>44</td>
<td>D</td>
<td>68</td>
<td>[</td>
<td>91</td>
</tr>
<tr>
<td>/</td>
<td>47</td>
<td>E</td>
<td>69</td>
<td>]</td>
<td>93</td>
</tr>
<tr>
<td>*</td>
<td>42</td>
<td>J</td>
<td>74</td>
<td>√</td>
<td>4138</td>
</tr>
<tr>
<td>-</td>
<td>45</td>
<td>O</td>
<td>79</td>
<td>VAR-LNK</td>
<td>4141</td>
</tr>
<tr>
<td>+</td>
<td>43</td>
<td>U</td>
<td>85</td>
<td>CHAR</td>
<td>4139</td>
</tr>
<tr>
<td>Key Modifier</td>
<td>Modifier</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>2nd</td>
<td>α</td>
<td>alpha</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENTER</td>
<td>CR 13</td>
<td>CR 13</td>
<td>ENTRY 4109</td>
<td>Approx 8205</td>
<td>CR 13</td>
</tr>
<tr>
<td>STO</td>
<td>STO</td>
<td>258 P 80</td>
<td>RCL 4354</td>
<td>@ 64</td>
<td>p 112</td>
</tr>
<tr>
<td>-</td>
<td>61 A 65</td>
<td>· 39</td>
<td>≠ 157</td>
<td>a 97</td>
<td></td>
</tr>
<tr>
<td>EE</td>
<td>EE</td>
<td>149 K 75</td>
<td>← 159</td>
<td>SYMB 8341</td>
<td>k 107</td>
</tr>
<tr>
<td>-</td>
<td>173 SPACE 32</td>
<td>ANS 4372</td>
<td>8365</td>
<td>SPACE 32</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>46 W 87</td>
<td>&gt; 62</td>
<td>≥ 158</td>
<td>w 119</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>48 V 86</td>
<td>&lt; 60</td>
<td>≤ 156</td>
<td>v 118</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>49 Q 81</td>
<td>“ 34</td>
<td>8241</td>
<td>q 113</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>50 R 50</td>
<td>\ 92</td>
<td>8242</td>
<td>r 114</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>51 S3 83</td>
<td>CUST 4147</td>
<td>8243</td>
<td>s 115</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>52 L 76</td>
<td>: 58</td>
<td>8244</td>
<td>l 108</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>53 M 77</td>
<td>MATH 4149</td>
<td>8245</td>
<td>m 109</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>54 N 78</td>
<td>MEM 4150</td>
<td>8246</td>
<td>n 110</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>55 G 71</td>
<td>j 4151</td>
<td>8247</td>
<td>g 103</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>56 H 72</td>
<td>d 4152</td>
<td>8248</td>
<td>h 104</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>57 I 73</td>
<td>; 59</td>
<td>8249</td>
<td>i 105</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Key Codes for Primary Keys
Table 2: Arrow Keys (including diagonal movement)

<table>
<thead>
<tr>
<th>Key</th>
<th>Normal</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>‡</td>
<td>338</td>
<td>16722</td>
<td>4434</td>
<td>8530</td>
<td>33106</td>
</tr>
<tr>
<td>⊳</td>
<td>340</td>
<td>16724</td>
<td>4436</td>
<td>8532</td>
<td>33108</td>
</tr>
<tr>
<td>⊵</td>
<td>344</td>
<td>16728</td>
<td>4440</td>
<td>8536</td>
<td>33112</td>
</tr>
<tr>
<td>⊴</td>
<td>337</td>
<td>16721</td>
<td>4433</td>
<td>8529</td>
<td>33105</td>
</tr>
<tr>
<td>⊳ and ‡</td>
<td>339</td>
<td>16723</td>
<td>4435</td>
<td>8531</td>
<td>33107</td>
</tr>
<tr>
<td>⊵ and ‡</td>
<td>342</td>
<td>16726</td>
<td>4438</td>
<td>8534</td>
<td>33110</td>
</tr>
<tr>
<td>⊵ and ⊳</td>
<td>345</td>
<td>16729</td>
<td>4441</td>
<td>8537</td>
<td>33113</td>
</tr>
<tr>
<td>⊳ and ⊴</td>
<td>348</td>
<td>16732</td>
<td>4444</td>
<td>8540</td>
<td>33116</td>
</tr>
</tbody>
</table>

Table 3: Greek Letters (prefixed by "[ ]")

<table>
<thead>
<tr>
<th>Keys</th>
<th>Second modifier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>alpha</td>
</tr>
<tr>
<td></td>
<td>f</td>
</tr>
<tr>
<td>Assoc.</td>
<td>Value</td>
</tr>
<tr>
<td>☙ [A]</td>
<td>α</td>
</tr>
<tr>
<td>☙ [B]</td>
<td>β</td>
</tr>
<tr>
<td>☙ [D]</td>
<td>δ</td>
</tr>
<tr>
<td>☙ [E]</td>
<td>ε</td>
</tr>
<tr>
<td>☙ [F]</td>
<td>φ</td>
</tr>
<tr>
<td>☙ [G]</td>
<td>γ</td>
</tr>
<tr>
<td>☙ [L]</td>
<td>λ</td>
</tr>
<tr>
<td>☙ [M]</td>
<td>μ</td>
</tr>
<tr>
<td>☙ [P]</td>
<td>π</td>
</tr>
<tr>
<td>☙ [R]</td>
<td>ρ</td>
</tr>
<tr>
<td>☙ [S]</td>
<td>σ</td>
</tr>
<tr>
<td>☙ [T]</td>
<td>τ</td>
</tr>
<tr>
<td>☙ [W]</td>
<td>ω</td>
</tr>
<tr>
<td>☙ [X]</td>
<td>ξ</td>
</tr>
<tr>
<td>☙ [Ψ]</td>
<td>ψ</td>
</tr>
<tr>
<td>☙ ζ</td>
<td>135</td>
</tr>
</tbody>
</table>

Voyage™ 200 Key Codes

The `getKey()` function returns a value that corresponds to the last key pressed, according to the tables shown in this section. For example, if your program contains a `getKey()` function, pressing 2nd f1 will return a value of 268.
<table>
<thead>
<tr>
<th>Key</th>
<th>Modifier</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>F1</strong></td>
<td>F1</td>
</tr>
<tr>
<td><strong>F2</strong></td>
<td>F2</td>
</tr>
<tr>
<td><strong>F3</strong></td>
<td>F3</td>
</tr>
<tr>
<td><strong>F4</strong></td>
<td>F4</td>
</tr>
<tr>
<td><strong>F5</strong></td>
<td>F5</td>
</tr>
<tr>
<td><strong>F6</strong></td>
<td>F6</td>
</tr>
<tr>
<td><strong>F7</strong></td>
<td>F7</td>
</tr>
<tr>
<td><strong>F8</strong></td>
<td>F8</td>
</tr>
<tr>
<td><strong>MODE</strong></td>
<td>MODE</td>
</tr>
<tr>
<td><strong>CLEAR</strong></td>
<td>CLEAR</td>
</tr>
<tr>
<td><strong>LN</strong></td>
<td>LN</td>
</tr>
<tr>
<td><strong>ESC</strong></td>
<td>ESC</td>
</tr>
<tr>
<td><strong>APPS</strong></td>
<td>APPS</td>
</tr>
<tr>
<td><strong>ENTER</strong></td>
<td>CR</td>
</tr>
<tr>
<td><strong>SIN</strong></td>
<td>SIN</td>
</tr>
<tr>
<td><strong>COS</strong></td>
<td>COS</td>
</tr>
<tr>
<td><strong>TAN</strong></td>
<td>TAN</td>
</tr>
<tr>
<td>▲</td>
<td>^</td>
</tr>
<tr>
<td>▼</td>
<td>(</td>
</tr>
<tr>
<td>}</td>
<td>)</td>
</tr>
<tr>
<td>,</td>
<td>,</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>STO</td>
<td>STO</td>
</tr>
<tr>
<td>SPACE</td>
<td>32</td>
</tr>
<tr>
<td>=</td>
<td>61</td>
</tr>
<tr>
<td>BS</td>
<td>257</td>
</tr>
<tr>
<td>Key</td>
<td>Modifier</td>
</tr>
<tr>
<td>-----</td>
<td>----------</td>
</tr>
<tr>
<td>None</td>
<td>1</td>
</tr>
<tr>
<td>[θ]</td>
<td>θ</td>
</tr>
<tr>
<td>( )</td>
<td>-</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>A</td>
<td>a</td>
</tr>
<tr>
<td>B</td>
<td>b</td>
</tr>
<tr>
<td>C</td>
<td>c</td>
</tr>
<tr>
<td>D</td>
<td>d</td>
</tr>
<tr>
<td>E</td>
<td>e</td>
</tr>
<tr>
<td>F</td>
<td>f</td>
</tr>
<tr>
<td>G</td>
<td>g</td>
</tr>
<tr>
<td>H</td>
<td>h</td>
</tr>
<tr>
<td>I</td>
<td>i</td>
</tr>
<tr>
<td>J</td>
<td>j</td>
</tr>
<tr>
<td>K</td>
<td>k</td>
</tr>
<tr>
<td>L</td>
<td>l</td>
</tr>
<tr>
<td>M</td>
<td>m</td>
</tr>
<tr>
<td>N</td>
<td>n</td>
</tr>
<tr>
<td>O</td>
<td>o</td>
</tr>
<tr>
<td>P</td>
<td>p</td>
</tr>
<tr>
<td>Q</td>
<td>q</td>
</tr>
<tr>
<td>R</td>
<td>r</td>
</tr>
<tr>
<td>Key</td>
<td>Modifier</td>
</tr>
<tr>
<td>-----</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td>None</td>
</tr>
<tr>
<td>S</td>
<td>s</td>
</tr>
<tr>
<td>T</td>
<td>t</td>
</tr>
<tr>
<td>U</td>
<td>u</td>
</tr>
<tr>
<td>V</td>
<td>v</td>
</tr>
<tr>
<td>W</td>
<td>w</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
</tr>
<tr>
<td>Y</td>
<td>y</td>
</tr>
<tr>
<td>Z</td>
<td>z</td>
</tr>
</tbody>
</table>

Key Modifier

<table>
<thead>
<tr>
<th>None</th>
<th>1st</th>
<th>2nd</th>
<th>POP</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>s</td>
<td>115</td>
<td>S</td>
</tr>
<tr>
<td>T</td>
<td>t</td>
<td>116</td>
<td>T</td>
</tr>
<tr>
<td>U</td>
<td>u</td>
<td>117</td>
<td>U</td>
</tr>
<tr>
<td>V</td>
<td>v</td>
<td>118</td>
<td>V</td>
</tr>
<tr>
<td>W</td>
<td>w</td>
<td>119</td>
<td>W</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td>120</td>
<td>X</td>
</tr>
<tr>
<td>Y</td>
<td>y</td>
<td>121</td>
<td>Y</td>
</tr>
<tr>
<td>Z</td>
<td>z</td>
<td>122</td>
<td>Z</td>
</tr>
</tbody>
</table>
### Table 2: Arrow Keys (including diagonal movement)

<table>
<thead>
<tr>
<th>Key</th>
<th>Normal</th>
<th>후</th>
<th>2nd</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>◄</td>
<td>338</td>
<td>16722</td>
<td>4434</td>
<td>8530</td>
<td>33106</td>
</tr>
<tr>
<td>◄</td>
<td>340</td>
<td>16724</td>
<td>4436</td>
<td>8532</td>
<td>33108</td>
</tr>
<tr>
<td>◄</td>
<td>344</td>
<td>16728</td>
<td>4440</td>
<td>8536</td>
<td>33112</td>
</tr>
<tr>
<td>◄</td>
<td>337</td>
<td>16721</td>
<td>4433</td>
<td>8529</td>
<td>33105</td>
</tr>
<tr>
<td>◄ and ◄</td>
<td>339</td>
<td>16723</td>
<td>4435</td>
<td>8531</td>
<td>33107</td>
</tr>
<tr>
<td>◄ and ◄</td>
<td>342</td>
<td>16726</td>
<td>4438</td>
<td>8534</td>
<td>33110</td>
</tr>
<tr>
<td>◄ and ◄</td>
<td>345</td>
<td>16729</td>
<td>4441</td>
<td>8537</td>
<td>33113</td>
</tr>
<tr>
<td>◄ and ◄</td>
<td>348</td>
<td>16732</td>
<td>4444</td>
<td>8540</td>
<td>33116</td>
</tr>
</tbody>
</table>

*Note: The Grab (.pause) modifier only affects the arrow keys.*

### Table 3: Grave Accent Letters (prefixed by 2nd A)

<table>
<thead>
<tr>
<th>Key</th>
<th>Assoc.</th>
<th>Normal</th>
<th>턵</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>à</td>
<td>224</td>
<td>192</td>
</tr>
<tr>
<td>E</td>
<td>è</td>
<td>232</td>
<td>200</td>
</tr>
<tr>
<td>I</td>
<td>i</td>
<td>236</td>
<td>204</td>
</tr>
<tr>
<td>O</td>
<td>ò</td>
<td>242</td>
<td>210</td>
</tr>
<tr>
<td>U</td>
<td>ü</td>
<td>249</td>
<td>217</td>
</tr>
</tbody>
</table>

### Table 4: Cedilla Letters (prefixed by 2nd C)

<table>
<thead>
<tr>
<th>Key</th>
<th>Assoc.</th>
<th>Normal</th>
<th>턵</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>ç</td>
<td>231</td>
<td>199</td>
</tr>
</tbody>
</table>
### Table 5: Acute Accent Letters (prefixed by \textsuperscript{2E})

<table>
<thead>
<tr>
<th>Key</th>
<th>Assoc.</th>
<th>Normal</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>á</td>
<td>225</td>
<td>193</td>
</tr>
<tr>
<td>E</td>
<td>é</td>
<td>233</td>
<td>201</td>
</tr>
<tr>
<td>I</td>
<td>í</td>
<td>237</td>
<td>205</td>
</tr>
<tr>
<td>O</td>
<td>ó</td>
<td>243</td>
<td>211</td>
</tr>
<tr>
<td>U</td>
<td>ú</td>
<td>250</td>
<td>218</td>
</tr>
<tr>
<td>Y</td>
<td>ý</td>
<td>253</td>
<td>221</td>
</tr>
</tbody>
</table>

### Table 6: Greek Letters (prefixed by \textsuperscript{2G})

<table>
<thead>
<tr>
<th>Key</th>
<th>Assoc.</th>
<th>Normal</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>α</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>β</td>
<td>129</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>δ</td>
<td>133</td>
<td>132</td>
</tr>
<tr>
<td>E</td>
<td>ε</td>
<td>134</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>φ</td>
<td>145</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>γ</td>
<td>131</td>
<td>130</td>
</tr>
<tr>
<td>L</td>
<td>λ</td>
<td>137</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>µ</td>
<td>181</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>π</td>
<td>140</td>
<td>139</td>
</tr>
<tr>
<td>R</td>
<td>ρ</td>
<td>141</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>σ</td>
<td>143</td>
<td>142</td>
</tr>
<tr>
<td>T</td>
<td>τ</td>
<td>144</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>ω</td>
<td>148</td>
<td>147</td>
</tr>
<tr>
<td>X</td>
<td>ξ</td>
<td>138</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>ψ</td>
<td>146</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>ζ</td>
<td>135</td>
<td></td>
</tr>
</tbody>
</table>
Table 7: Tilde Letters (prefixed by 2nd N)

<table>
<thead>
<tr>
<th>Key</th>
<th>Assoc.</th>
<th>Normal</th>
<th>( \text{N} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>ŋ</td>
<td>241</td>
<td>209</td>
</tr>
<tr>
<td>O</td>
<td>ô</td>
<td>245</td>
<td></td>
</tr>
</tbody>
</table>

Table 8: Caret Letters (prefixed by 2nd O)

<table>
<thead>
<tr>
<th>Key</th>
<th>Assoc.</th>
<th>Normal</th>
<th>( \text{O} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>å</td>
<td>226</td>
<td>194</td>
</tr>
<tr>
<td>E</td>
<td>ê</td>
<td>234</td>
<td>202</td>
</tr>
<tr>
<td>I</td>
<td>ï</td>
<td>238</td>
<td>206</td>
</tr>
<tr>
<td>O</td>
<td>ô</td>
<td>244</td>
<td>212</td>
</tr>
<tr>
<td>U</td>
<td>ū</td>
<td>251</td>
<td>219</td>
</tr>
</tbody>
</table>

Table 9: Umlaut Letters (prefixed by 2nd U)

<table>
<thead>
<tr>
<th>Key</th>
<th>Assoc.</th>
<th>Normal</th>
<th>( \text{U} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ä</td>
<td>228</td>
<td>196</td>
</tr>
<tr>
<td>E</td>
<td>ë</td>
<td>235</td>
<td>203</td>
</tr>
<tr>
<td>I</td>
<td>ï</td>
<td>239</td>
<td>207</td>
</tr>
<tr>
<td>O</td>
<td>ò</td>
<td>246</td>
<td>214</td>
</tr>
<tr>
<td>U</td>
<td>ū</td>
<td>252</td>
<td>220</td>
</tr>
<tr>
<td>Y</td>
<td>ſ</td>
<td>255</td>
<td></td>
</tr>
</tbody>
</table>
Entering Complex Numbers

You can enter complex numbers in the polar form \((r \angle \theta)\), where \(r\) is the magnitude and \(\theta\) is the angle, or polar form \(r e^{i \theta}\). You can also enter complex numbers in rectangular form \(a + bi\).

Overview of Complex Numbers

A complex number has real and imaginary components that identify a point in the complex plane. These components are measured along the real and imaginary axes, which are similar to the \(x\) and \(y\) axes in the real plane.

The point can be expressed in rectangular form or in either of two polar forms.

The \(i\) symbol represents the imaginary number \(\sqrt{-1}\).

As shown below, the form that you can enter depends on the current Angle mode.

<table>
<thead>
<tr>
<th>You can use the form:</th>
<th>When the Angle mode setting is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a + bi)</td>
<td>Radian, Degree or Gradian</td>
</tr>
<tr>
<td>(r e^{i \theta})</td>
<td>Radian only</td>
</tr>
<tr>
<td>((r \angle \theta))</td>
<td>Radian, Degree or Gradian</td>
</tr>
</tbody>
</table>

(In Degree or Gradian angle mode, this form causes a Domain error.)

Use the following methods to enter a complex number.

<table>
<thead>
<tr>
<th>To enter the:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangular form</td>
<td>Substitute the applicable values or variable names for (a) and (b).</td>
</tr>
<tr>
<td>(a + bi)</td>
<td>(a \div b \text{ [(i)]} )</td>
</tr>
<tr>
<td>(b \text{ [(i)]} )</td>
<td>Note: To get the (i) symbol, press (2\text{[(i)]}). Do not simply type an alphabetic (i).</td>
</tr>
</tbody>
</table>

For example:
Polar form

You can enter a complex number at any time, regardless of the Complex Format mode setting. However, the mode setting determines how results are displayed.

TI-89 Titanium:

\[
\alpha \begin{bmatrix} R \end{bmatrix} \begin{bmatrix} a+i \end{bmatrix} 2\text{nd} \begin{bmatrix} 0 \end{bmatrix} \begin{bmatrix} \theta \end{bmatrix}
\]

or

\[
\begin{bmatrix} \alpha \end{bmatrix} \begin{bmatrix} R \end{bmatrix} 2\text{nd} \begin{bmatrix} (\angle \theta \end{bmatrix} \begin{bmatrix} \theta \end{bmatrix}
\]

Important: Do not use the \( re^{i\theta} \) polar form in Degree angle mode. It will cause a Domain error.

Note: To get the \( e \) symbol, press:

- TI-89 Titanium: \( \boxed{[a+i]} \)
- Voyage™ 200: \( 2\boxed{s} \)

Do not simply type an alphabetic \( e \).

Tip: To get the \( \angle \) symbol, press \( 2\boxed{\theta} \).

Tip: To enter \( \theta \) in degrees for \( r \angle \theta \), you can type a \( ^\circ \) symbol (such as 45\(^\circ\)). To get the \( ^\circ \) symbol, press \( 2\boxed{\theta} \). You should not use degrees or Gradian for \( re^{i\theta} \).

Voyage™ 200:

\[
\begin{bmatrix} R \end{bmatrix} 2\text{nd} \begin{bmatrix} a+i \end{bmatrix} \begin{bmatrix} \theta \end{bmatrix} \begin{bmatrix} \theta \end{bmatrix}
\]

or

\[
\begin{bmatrix} \alpha \end{bmatrix} \begin{bmatrix} R \end{bmatrix} 2\text{nd} \begin{bmatrix} \theta \end{bmatrix} \begin{bmatrix} \theta \end{bmatrix}
\]

For example:

\[
\begin{array}{c}
\frac{4}{\text{e}^{-\pi}} \\
\frac{2}{\text{e}^{3\pi}i} \\
\frac{2}{\text{e}^{2\pi}i}
\end{array}
\]

Results are shown in rectangular form, but you can select polar form.

---

**Complex Format Mode for Displaying Results**

Use \( \boxed{\text{MODE}} \) to set the Complex Format mode to one of three settings.

You can enter a complex number at any time, regardless of the Complex Format mode setting. However, the mode setting determines how results are displayed.

---

**If Complex Format is:**

- **The TI-89 Titanium / Voyage™ 200:**
Regardless of the Complex Format mode setting, variables that have no stored value and those that do not end with an underscore (_) are treated as real numbers. To perform complex symbolic analysis, you can use either of the following methods to set up a complex variable.

**Method 1:**

Use an underscore _ (TI-89 Titanium: \(\text{\textasciitilde}\) \(\text{\textasciitilde}\), Voyage™ 200 \(\text{\textasciitilde}\) \(\text{\textasciitilde}\)) as the last character in the variable name to designate a complex variable. For example:

```plaintext
z_ is treated as a complex variable if it does not have a stored value.
```

**Method 2:** Store an unreal value into any variable. For example:

```plaintext
x+y\text{i} \rightarrow z
Then z is treated as a complex variable.
```

**Note:** For best results in calculations such as `cSolve()` and `cZeros()`, use Method 1.

### Using Complex Variables in Symbolic Calculations

Radian angle mode is recommended for complex number calculations. Internally, the TI-89 Titanium / Voyage™ 200 converts all entered trig values to radians, but it does not convert values for exponential, logarithmic, or hyperbolic functions.

In Degree angle mode, complex identities such as \(e^{(\text{i} \cdot \theta)} = \cos(\theta) + \text{i} \sin(\theta)\) are not generally true because the values for \(\cos\) and \(\sin\) are converted to radians, while those for \(e^{\text{n}}()\) are not. For example, \(e^{(\text{i} \cdot 45)} = \cos(45) + \text{i} \sin(45)\) is treated internally as \(e^{(\text{i} \cdot 45)} = \cos(\pi/4) + \text{i} \sin(\pi/4)\). Complex identities are always true in Radian angle mode.

**Note:** If you use Degree angle mode, you must make polar entries in the form \((r \angle \theta)\). In Degree or Gradian angle mode, an \(r\text{e}\) entry causes an error.
Accuracy Information

To maximize accuracy, the TI-89 Titanium / Voyage™ 200 carries more digits internally than it displays.

Computational Accuracy

Floating-point (decimal) values in memory are stored using up to 14 digits with a 3-digit exponent.

- For min and max Window variables (xmin, xmax, ymin, ymax, etc.), you can store values using up to 12 digits. Other Window variables use 14 digits.
- When a floating-point value is displayed, the displayed value is rounded as specified by the applicable mode settings (Display Digits, Exponential Format, etc.), with a maximum of 12 digits and a 3-digit exponent.
- RegEQ displays up to 14-digit coefficients.

Integer values in memory are stored using up to 614 digits.

Graphing Accuracy

The Window variable xmin is the center of the leftmost pixel used, and xmax is the center of the rightmost pixel used. Δx is the distance between the centers of two horizontally adjacent pixels.

- Δx is calculated as (xmax − xmin) / (# of x pixels − 1).
- If Δx is entered from the Home screen or a program, xmax is calculated as xmin + Δx * (# of x pixels − 1).

Note: For a table that lists the number of pixels in a full screen or split screen, refer to “Setting and Exiting the Split Screen Mode” in Split Screens.

The Window variable ymin is the center of the bottom pixel used, and ymax is the center of the top pixel used. Δy is the distance between the centers of two vertically adjacent pixels.

- Δy is calculated as (ymax − ymin) / (# of y pixels − 1).
- If Δy is entered from the Home screen or a program, ymax is calculated as ymin + Δy * (# of y pixels − 1).

Cursor coordinates are displayed as eight characters (which may include a negative sign, decimal point, and exponent). The coordinate values (xc, yc, zc, etc.) are updated with a maximum of 12-digit accuracy.

System Variables and Reserved Names

This section lists the names of system variables and reserved function names that are used by the TI-89 Titanium / Voyage™ 200. Only those system variables and reserved function names that are identified by an asterisk (*) can be deleted by using DelVar var on the entry line.
Graph

\[ y_1(x) - y_{99}(x)^* \quad y_1'(t) - y_{99}'(t)^* \quad y_i(t) - y_{99}(t)^* \quad r_1(q) - r_{99}(q)^* \]

\[ x_1 - x_{99}^* \quad y_1 - y_{99}^* \quad x_{t1}(t) - x_{t99}(t)^* \quad y_{t1}(t) - y_{t99}(t)^* \]

\[ \Delta x \quad \Delta y \quad z_{\min} \quad z_{\max} \]

\[ z_{\min} \quad z_{\max} \quad z_{\text{step}} \]

\[ t_{\min} \quad t_{\max} \quad t_{\text{step}} \quad t_0 \]

\[ n_{\text{curves}} \quad \text{dftol} \quad \text{dtim} \]

\[ n_{\text{start}} \quad n_{\text{end}} \quad n_{\text{max}} \]

\[ \text{sysMath} \]

Graph Zoom

\[ z_{\text{min}} \quad z_{\text{max}} \quad z_{\text{scl}} \quad z_{\text{grid}} \]

\[ y_{\text{min}} \quad y_{\text{max}} \quad y_{\text{scl}} \quad y_{\text{grid}} \]

\[ z_{\text{res}} \quad z_{\text{min}} \quad z_{\text{max}} \quad z_{\text{step}} \]

\[ z_{\text{max}} \quad z_{\text{min}} \quad z_{\text{step}} \quad z_{\text{plott}} \]

\[ z_{\text{grid}} \]

Statistics

\[ \bar{x} \quad \bar{y} \quad \Sigma x \quad \sigma x \]

\[ \Sigma x^2 \quad \Sigma x y \quad \Sigma y \quad \sigma y \]

\[ \Sigma y^2 \quad \text{corr} \quad \text{maxX} \quad \text{maxY} \]

\[ \text{medStat} \quad \text{medx1} \quad \text{medx2} \quad \text{medx3} \]

\[ \text{medy1} \quad \text{medy2} \quad \text{medy3} \quad \text{minX} \]

\[ \text{minY} \quad n_{\text{Stat}} \quad q1 \quad q3 \]

\[ \text{regCoef} \quad \text{regEq}(x)^* \quad \text{seed1} \quad \text{seed2} \]

\[ S_x \quad S_y \quad R^2 \]
This section describes the Equation Operating System (EOS™) that is used by the TI-89 Titanium / Voyage™ 200. Numbers, variables, and functions are entered in a simple, straightforward sequence. EOS evaluates expressions and equations using parenthetical grouping and according to the priorities described below.

### Order of Evaluation

<table>
<thead>
<tr>
<th>Level</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Parentheses ( ), brackets [ ], braces { }</td>
</tr>
<tr>
<td>2</td>
<td>Indirection (#)</td>
</tr>
<tr>
<td>3</td>
<td>Function calls</td>
</tr>
<tr>
<td>4</td>
<td>Post operators: degrees-minutes-seconds (°',&quot;)&quot;, factorial (!), percentage (%), radian (ô), subscript ([ ]), transpose (î)</td>
</tr>
<tr>
<td>5</td>
<td>Exponentiation, power operator (^)</td>
</tr>
<tr>
<td>6</td>
<td>Negation ( ¬)</td>
</tr>
<tr>
<td>7</td>
<td>String concatenation (&amp;)</td>
</tr>
<tr>
<td>8</td>
<td>Multiplication (•), division (/)</td>
</tr>
<tr>
<td>9</td>
<td>Addition (+), subtraction (−)</td>
</tr>
<tr>
<td>10</td>
<td>Equality relations: equal (=), not equal (≠ or /=), less than (&lt;), less than or equal (≤ or &lt;=), greater than (&gt;) , greater than or equal (≥ or &gt;=)</td>
</tr>
<tr>
<td>11</td>
<td>Logical not</td>
</tr>
<tr>
<td>12</td>
<td>Logical and</td>
</tr>
<tr>
<td>13</td>
<td>Logical or, exclusive logical xor</td>
</tr>
</tbody>
</table>
Parentheses, Brackets, and Braces

All calculations inside a pair of parentheses, brackets, or braces are evaluated first. For example, in the expression 4(1+2), EOS first evaluates the portion of the expression inside the parentheses, 1+2, and then multiplies the result, 3, by 4.

The number of opening and closing parentheses, brackets, and braces must be the same within an expression or equation. If not, an error message is displayed that indicates the missing element. For example, (1+2)/(3+4 will display the error message “Missing ”.

Note: Because the TI-89 Titanium / Voyage™ 200 allows you to define your own functions, a variable name followed by an expression in parentheses is considered a “function call” instead of implied multiplication. For example a(b+c) is the function a evaluated by b+c. To multiply the expression b+c by the variable a, use explicit multiplication: a*(b+c).

Indirection

The indirection operator (#) converts a string to a variable or function name. For example, #("x"&"y"&"z") creates the variable name xyz. Indirection also allows the creation and modification of variables from inside a program. For example, if 10→r and "r"→s1, then #s1=10.

Post Operators

Post operators are operators that come directly after an argument, such as 5!, 25%, or 60° 15' 45". Arguments followed by a post operator are evaluated at the fourth priority level. For example, in the expression 4^3!, 3! is evaluated first. The result, 6, then becomes the exponent of 4 to yield 4096.

Exponentiation

Exponentiation (^) and element-by-element exponentiation (.^) are evaluated from right to left. For example, the expression 2^3^2 is evaluated the same as 2^(3^2) to produce 512. This is different from (2^3)^2, which is 64.

Negation

To enter a negative number, press · followed by the number. Post operations and exponentiation are performed before negation. For example, the result of -x^2 is a negative number, and -9^2 = -81. Use parentheses to square a negative number such as (-9)^2 to produce 81. Note also that negative 5 (-5) is different from minus 5 (-5), and -3! evaluates as -(3!).

Constraint (|)

The argument following the “with” (|) operator provides a set of constraints that affect the evaluation of the argument preceding the “with” operator.
Regression Formulas

This section describes how the statistical regressions are calculated.

Least-Squares Algorithm

Most of the regressions use non-linear recursive least-squares techniques to optimize the following cost function, which is the sum of the squares of the residual errors:

$$J = \sum_{i=1}^{N} [\text{residualExpression}]^2$$

where: residualExpression is in terms of $x_i$ and $y_i$

- $x_i$ is the independent variable list
- $y_i$ is the dependent variable list
- $N$ is the dimension of the lists

This technique attempts to recursively estimate the constants in the model expression to make $J$ as small as possible.

For example, $y = a \sin(bx+c)+d$ is the model equation for SinReg. So its residual expression is:

$$a \sin(bx+c)+d - y_i$$

For SinReg, therefore, the least-squares algorithm finds the constants $a$, $b$, $c$, and $d$ that minimize the function:

$$J = \sum_{i=1}^{N} [a \sin(bx_i+c)+d - y_i]^2$$

Regressions

<table>
<thead>
<tr>
<th>Regression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CubicReg</td>
<td>Uses the least-squares algorithm to fit the third-order polynomial: $y=ax^3+bx^2+cx+d$ For four data points, the equation is a polynomial fit; for five or more, it is a polynomial regression. At least four data points are required.</td>
</tr>
<tr>
<td>ExpReg</td>
<td>Uses the least-squares algorithm and transformed values $x$ and $\ln(y)$ to fit the model equation: $y=ab^x$</td>
</tr>
<tr>
<td>LinReg</td>
<td>Uses the least-squares algorithm to fit the model equation: $y=ax+b$ where $a$ is the slope and $b$ is the $y$-intercept.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LnReg</td>
<td>Uses the least-squares algorithm and transformed values $\ln(x)$ and $y$ to fit the model equation:</td>
</tr>
<tr>
<td></td>
<td>$y=a+b\ln(x)$</td>
</tr>
<tr>
<td>Logistic</td>
<td>Uses the least-squares algorithm to fit the model equation:</td>
</tr>
<tr>
<td></td>
<td>$y=a/(1+b\cdot e^{(c\cdot x)})+d$</td>
</tr>
<tr>
<td>MedMed</td>
<td>Uses the median-median line (resistant line) technique to calculate summary points $x_1$, $y_1$, $x_2$,</td>
</tr>
<tr>
<td></td>
<td>$x_3$, and $y_3$, and fits the model equation:</td>
</tr>
<tr>
<td></td>
<td>$y=ax+b$</td>
</tr>
<tr>
<td></td>
<td>where $a$ is the slope and $b$ is the y-intercept.</td>
</tr>
<tr>
<td>PowerReg</td>
<td>Uses the least-squares algorithm and transformed values $\ln(x)$ and $\ln(y)$ to fit the model equation:</td>
</tr>
<tr>
<td></td>
<td>$y=ax^b$</td>
</tr>
<tr>
<td>QuadReg</td>
<td>Uses the least-squares algorithm to fit the second-order polynomial:</td>
</tr>
<tr>
<td></td>
<td>$y=ax^2+bx+c$</td>
</tr>
<tr>
<td></td>
<td>For three data points, the equation is a polynomial fit; for four or more, it is a polynomial regression.</td>
</tr>
<tr>
<td></td>
<td>At least three data points are required.</td>
</tr>
<tr>
<td>QuartReg</td>
<td>Uses the least-squares algorithm to fit the fourth-order polynomial:</td>
</tr>
<tr>
<td></td>
<td>$y=ax^4+bx^3+cx^2+dx+e$</td>
</tr>
<tr>
<td></td>
<td>For five data points, the equation is a polynomial fit; for six or more, it is a polynomial regression.</td>
</tr>
<tr>
<td></td>
<td>At least five data points are required.</td>
</tr>
<tr>
<td>SinReg</td>
<td>Uses the least-squares algorithm to fit the model equation:</td>
</tr>
<tr>
<td></td>
<td>$y=a\sin(bx+c)+d$</td>
</tr>
</tbody>
</table>
Contour Levels and Implicit Plot Algorithm

Contours are calculated and plotted by the following method. An implicit plot is the same as a contour, except that an implicit plot is for the z=0 contour only.

Algorithm

Based on your x and y Window variables, the distance between xmin and xmax and between ymin and ymax is divided into a number of grid lines specified by xgrid and ygrid. These grid lines intersect to form a series of rectangles.

For each rectangle, the equation is evaluated at each of the four corners (also called vertices or grid points) and an average value (E) is calculated:

\[ E = \frac{z_1 + z_2 + z_3 + z_4}{4} \]

The E value is treated as the value of the equation at the center of the rectangle.

For each specified contour value (Ci)

- At each of the five points shown to the right, the difference between the point’s z value and the contour value is calculated.

- A sign change between any two adjacent points implies that a contour crosses the line that joins those two points. Linear interpolation is used to approximate where the zero crosses the line.

- Within the rectangle, any zero crossings are connected with straight lines.

- This process is repeated for each contour value.

Each rectangle in the grid is treated similarly.

Runge-Kutta Method

For Runge-Kutta integrations of ordinary differential equations, the TI-89 Titanium / Voyage™ 200 uses the Bogacki-Shampine 3(2) formula as found in the journal *Applied Math Letters*, 2 (1989), pp. 1–9.
The Bogacki-Shampine 3(2) formula provides a result of 3rd-order accuracy and an error estimate based on an embedded 2nd-order formula. For a problem of the form:

\[ y' = f(x, y) \]

and a given step size \( h \), the Bogacki-Shampine formula can be written:

\[ F_1 = f(x_n, y_n) \]

\[ F_2 = f\left(x_n + h \frac{1}{2}, y_n + h \frac{1}{2} F_1\right) \]

\[ F_3 = f\left(x_n + h \frac{3}{4}, y_n + h \frac{3}{4} F_2\right) \]

\[ y_{n+1} = y_n + h\left(\frac{2}{9} F_1 + \frac{1}{3} F_2 + \frac{4}{9} F_3\right) \]

\[ x_{n+1} = x_n + h \]

\[ F_4 = f\left(x_{n+1}, y_{n+1}\right) \]

\[ \text{errest} = h\left(\frac{5}{72} F_1 - \frac{1}{12} F_2 - \frac{1}{9} F_3 + \frac{1}{8} F_4\right) \]

The error estimate \( \text{errest} \) is used to control the step size automatically. For a thorough discussion of how this can be done, refer to *Numerical Solution of Ordinary Differential Equations* by L. F. Shampine (New York: Chapman & Hall, 1994).

The TI-89 Titanium / Voyage™ 200 software does not adjust the step size to land on particular output points. Rather, it takes the biggest steps that it can (based on the error tolerance \( \text{dtol} \)) and obtains results for \( x_n \leq x \leq x_{n+1} \) using the cubic interpolating polynomial passing through the point \((x_n, y_n)\) with slope \( F_1 \) and through \((x_{n+1}, y_{n+1})\) with slope \( F_4 \). The interpolant is efficient and provides results throughout the step that are just as accurate as the results at the ends of the step.
Battery Information

The TI-89 Titanium / Voyage™ 200 uses two types of batteries: four alkaline batteries, and a lithium battery as a backup for retaining memory while you change the alkaline batteries.

When to Replace the Batteries

As the alkaline batteries run down, the display will begin to dim (especially during calculations). To compensate for this, you will need to adjust the contrast to a higher setting. If you find it necessary to increase the contrast setting frequently, you will need to replace the alkaline batteries. To assist you, a BATT indicator (_battery_) will display in the status line area when the batteries have drained down to the point when you should replace them soon. When the BATT indicator is displayed in reverse text (_reverse_BATT_), you must replace the alkaline batteries immediately.

To avoid loss of data, do not remove the lithium battery unless four fresh alkaline batteries are installed. Replace the lithium backup battery about every three or four years.

Note: To avoid loss of information stored in memory, the TI-89 Titanium / Voyage 200 must be off. Do not remove the alkaline batteries and the lithium battery at the same time.

Effects of Replacing the Batteries

If you do not remove both types of batteries at the same time or allow them to run down completely, you can change either type of battery without losing anything in memory.

Battery Precautions

Take these precautions when replacing batteries:

- Do not leave batteries within the reach of children.
- Do not mix new and used batteries. Do not mix brands (or types within brands) of batteries.
- Do not mix rechargeable and non-rechargeable batteries.
- Install batteries according to polarity (+ and -) diagrams.
- Do not place non-rechargeable batteries in a battery recharger.
- Properly dispose of used batteries immediately.
- Do not incinerate or dismantle batteries.
Replacing the Alkaline Batteries in the TI-89 Titanium

1. If the TI-89 Titanium is on, turn it off (press 2nd [OFF]) to avoid loss of information stored in memory.
2. Slide the protective cover over the keyboard and place the device face down.
3. Push down on the battery cover latch, and then pull up to remove the cover.
4. Remove all four discharged AAA batteries.
5. Install four new AAA alkaline batteries, arranged according to the polarity (+ and -) diagram inside the battery compartment.
6. Replace the battery cover by inserting the two prongs into the two slots at the bottom of the battery compartment, and then push the cover until the latch snaps closed.

Replacing the Lithium Battery in the TI-89 Titanium

To replace the lithium backup battery, remove the battery cover and unscrew the tiny screw holding the BACK UP BATTERY cover in place.

Remove the old battery and install a new WR44SW or 303 battery, positive (+) side up. Replace the cover and the screw.

Replacing the Alkaline Batteries in the Voyage 200

1. If the Voyage™ 200 is on, turn it off (press 2nd [OFF]) to avoid loss of information stored in memory.
2. Slide the protective cover over the keyboard and place the device face down.
3. Press the notched battery cover and slide it off, away from the device.
4. Remove all four discharged AAA batteries.
5. Install four new AAA alkaline batteries, arranged according to the polarity (+ and -) diagram inside the battery compartment.
6. Slide the battery cover onto the device, prong side first. Gently push the cover until the prongs snap into place.
Replacing the Lithium Battery in the Voyage 200

To replace the lithium backup battery, remove the battery cover. Insert a blunt object into the circular indentation next to the battery. Gently place a finger on the lithium battery and pry the battery out.

Slide in a new CR1616 or CR1620 battery, positive (+) side up. Press firmly to snap the new lithium battery into place.
If you have difficulty operating the TI-89 Titanium / Voyage™ 200, the following suggestions may help you correct the problem.

<table>
<thead>
<tr>
<th>If:</th>
<th>Suggested action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>You cannot see anything on the display.</td>
<td>Press $\downarrow$ to darken or $\uparrow$ to lighten the display contrast.</td>
</tr>
<tr>
<td>The BATT indicator is displayed.</td>
<td>Replace the batteries. If BATT is displayed in reverse text (BATT), replace the batteries as soon as possible.</td>
</tr>
<tr>
<td>The BUSY indicator is displayed.</td>
<td>A calculation is in progress. If you want to stop the calculation, press ON.</td>
</tr>
<tr>
<td>The PAUSE indicator is displayed.</td>
<td>A graph or program is paused and the TI-89 Titanium / Voyage 200 is waiting for input; press ENTER.</td>
</tr>
<tr>
<td>An error message is displayed.</td>
<td>Refer to the list of error messages in this module. Press ESC to clear.</td>
</tr>
<tr>
<td>The TI-89 Titanium / Voyage 200 does not appear to be working properly.</td>
<td>Press ESC several times to exit any menu or dialog box and to return the cursor to the entry line. — or — Be sure that the batteries are installed properly and that they are fresh.</td>
</tr>
<tr>
<td>If:</td>
<td>Suggested action:</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>The TI-89 Titanium appears to be “locked up” and will not respond to keyboard input.</td>
<td>The following action clears RAM. This erases all data, programs, and user-defined variables, functions, or folders. Press and hold 0, 0, and 2. Then press and release 2.</td>
</tr>
<tr>
<td></td>
<td>The following action clears RAM and Flash ROM. This erases all data, programs, user-defined variables, functions, folders, Flash applications, and the user data archive.</td>
</tr>
<tr>
<td>1. Remove one of the four AAA batteries.</td>
<td>1. Remove one of the four AAA batteries.</td>
</tr>
<tr>
<td>2. Press and hold 2 and 2 as you reinstall the battery.</td>
<td>2. Press and hold 2 and 2 as you reinstall the battery.</td>
</tr>
<tr>
<td>3. Continue holding 2 and 2 for five seconds before releasing.</td>
<td>3. Continue holding 2 and 2 for five seconds before releasing.</td>
</tr>
<tr>
<td>The Voyage™ 200 appears to be “locked up” and will not respond to keyboard input.</td>
<td>The following action clears RAM. This erases all data, programs, and user-defined variables, functions, or folders. Press and hold 2 and 2. Then press and release 2.</td>
</tr>
<tr>
<td></td>
<td>The following action clears RAM and Flash ROM. This erases all data, programs, user-defined variables, functions, folders, Flash applications, and the user data archive.</td>
</tr>
<tr>
<td>1. Remove one of the four AAA batteries.</td>
<td>1. Remove one of the four AAA batteries.</td>
</tr>
<tr>
<td>2. Press and hold 2 and 2 as you reinstall the battery.</td>
<td>2. Press and hold 2 and 2 as you reinstall the battery.</td>
</tr>
<tr>
<td>3. Continue holding 2 and 2 for five seconds before releasing.</td>
<td>3. Continue holding 2 and 2 for five seconds before releasing.</td>
</tr>
</tbody>
</table>
Appendix C: Programmer's Guide

The parameter/mode strings used in the `setMode()`, `getMode()`, `setGraph()`, and `setTable()` functions do not translate into other languages when used in a program. For example, when you write a program in the French Language mode then switch to the Italian Language mode, the program will produce an error. To avoid this error, you must substitute digits for the alpha characters. These digits operate in all languages. This appendix contains the digit substitutions for these strings.

The following examples show how to substitute digits in the `setMode()` function.

Example 1: A program using alpha parameter/mode strings:

```plaintext
setMode("Graph","Sequence")
```

Example 2: The same program, substituting digits for those strings:

```plaintext
setMode("1","4")
```
### setMode( ) and getMode( )

<table>
<thead>
<tr>
<th>Parameter/Mode Setting</th>
<th>Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>0</td>
</tr>
<tr>
<td>Graph</td>
<td>1</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>1</td>
</tr>
<tr>
<td>PARAMETRIC</td>
<td>2</td>
</tr>
<tr>
<td>POLAR</td>
<td>3</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>4</td>
</tr>
<tr>
<td>3D</td>
<td>5</td>
</tr>
<tr>
<td>DIFF EQUATIONS</td>
<td>6</td>
</tr>
<tr>
<td>DisplayDigits</td>
<td>2</td>
</tr>
<tr>
<td>FIX 0</td>
<td>1</td>
</tr>
<tr>
<td>FIX 1</td>
<td>2</td>
</tr>
<tr>
<td>FIX 2</td>
<td>3</td>
</tr>
<tr>
<td>FIX 3</td>
<td>4</td>
</tr>
<tr>
<td>FIX 4</td>
<td>5</td>
</tr>
<tr>
<td>FIX 5</td>
<td>6</td>
</tr>
<tr>
<td>FIX 6</td>
<td>7</td>
</tr>
<tr>
<td>FIX 7</td>
<td>8</td>
</tr>
<tr>
<td>FIX 8</td>
<td>9</td>
</tr>
<tr>
<td>Parameter/Mode Setting</td>
<td>Strings</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>FIX 9</td>
<td>10</td>
</tr>
<tr>
<td>FIX 10</td>
<td>11</td>
</tr>
<tr>
<td>FIX 11</td>
<td>12</td>
</tr>
<tr>
<td>FIX 12</td>
<td>13</td>
</tr>
<tr>
<td>FLOAT</td>
<td>14</td>
</tr>
<tr>
<td>FLOAT 1</td>
<td>15</td>
</tr>
<tr>
<td>FLOAT 2</td>
<td>16</td>
</tr>
<tr>
<td>FLOAT 3</td>
<td>17</td>
</tr>
<tr>
<td>FLOAT 4</td>
<td>18</td>
</tr>
<tr>
<td>FLOAT 5</td>
<td>19</td>
</tr>
<tr>
<td>FLOAT 6</td>
<td>20</td>
</tr>
<tr>
<td>FLOAT 7</td>
<td>21</td>
</tr>
<tr>
<td>FLOAT 8</td>
<td>22</td>
</tr>
<tr>
<td>FLOAT 9</td>
<td>23</td>
</tr>
<tr>
<td>FLOAT 10</td>
<td>24</td>
</tr>
<tr>
<td>FLOAT 11</td>
<td>25</td>
</tr>
<tr>
<td>FLOAT 12</td>
<td>26</td>
</tr>
<tr>
<td>Angle</td>
<td>3</td>
</tr>
<tr>
<td>RADIANS</td>
<td>1</td>
</tr>
<tr>
<td>Parameter/Mode Setting</td>
<td>Strings</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DEGREE</td>
<td>2</td>
</tr>
<tr>
<td>GRADIAN</td>
<td>3</td>
</tr>
<tr>
<td>Exponential Format</td>
<td>4</td>
</tr>
<tr>
<td>NORMAL</td>
<td>1</td>
</tr>
<tr>
<td>SCIENTIFIC</td>
<td>2</td>
</tr>
<tr>
<td>ENGINEERING</td>
<td>3</td>
</tr>
<tr>
<td>Complex Format</td>
<td>5</td>
</tr>
<tr>
<td>REAL</td>
<td>1</td>
</tr>
<tr>
<td>RECTANGULAR</td>
<td>2</td>
</tr>
<tr>
<td>POLAR</td>
<td>3</td>
</tr>
<tr>
<td>Vector Format</td>
<td>6</td>
</tr>
<tr>
<td>RECTANGULAR</td>
<td>1</td>
</tr>
<tr>
<td>CYLINDRICAL</td>
<td>2</td>
</tr>
<tr>
<td>SPHERICAL</td>
<td>3</td>
</tr>
<tr>
<td>Pretty Print</td>
<td>7</td>
</tr>
<tr>
<td>OFF</td>
<td>1</td>
</tr>
<tr>
<td>ON</td>
<td>2</td>
</tr>
<tr>
<td>SplitScreen</td>
<td>8</td>
</tr>
<tr>
<td>FULL</td>
<td>1</td>
</tr>
<tr>
<td>Parameter/Mode Setting</td>
<td>Strings</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>TOP-BOTTOM</td>
<td>2</td>
</tr>
<tr>
<td>LEFT-RIGHT</td>
<td>3</td>
</tr>
<tr>
<td>Split1App</td>
<td>9</td>
</tr>
<tr>
<td>(applications are not numbered)</td>
<td></td>
</tr>
<tr>
<td>Split2App</td>
<td>10</td>
</tr>
<tr>
<td>(applications are not numbered)</td>
<td></td>
</tr>
<tr>
<td>Number of Graphs</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter/Mode Setting</th>
<th>Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph 2</td>
<td>12</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>1</td>
</tr>
<tr>
<td>PARAMETRIC</td>
<td>2</td>
</tr>
<tr>
<td>POLAR</td>
<td>3</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>4</td>
</tr>
<tr>
<td>3D</td>
<td>5</td>
</tr>
<tr>
<td>DIFF_EQUATIONS</td>
<td>6</td>
</tr>
<tr>
<td>Split Screen Ratio</td>
<td>13</td>
</tr>
<tr>
<td>1:1</td>
<td>1</td>
</tr>
<tr>
<td>1:2</td>
<td>2</td>
</tr>
<tr>
<td>Parameter/Mode Setting</td>
<td>Strings</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>2:1</td>
<td>3</td>
</tr>
<tr>
<td>Exact/Approx</td>
<td>14</td>
</tr>
<tr>
<td>AUTO</td>
<td>1</td>
</tr>
<tr>
<td>EXACT</td>
<td>2</td>
</tr>
<tr>
<td>APPROXIMATE</td>
<td>3</td>
</tr>
<tr>
<td>Base</td>
<td>15</td>
</tr>
<tr>
<td>DEC</td>
<td>1</td>
</tr>
<tr>
<td>HEX</td>
<td>2</td>
</tr>
<tr>
<td>BIN</td>
<td>3</td>
</tr>
</tbody>
</table>
### setGraph( )

<table>
<thead>
<tr>
<th>Parameter/Mode Setting</th>
<th>Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinates 1</td>
<td></td>
</tr>
<tr>
<td>RECT</td>
<td>1</td>
</tr>
<tr>
<td>POLAR</td>
<td>2</td>
</tr>
<tr>
<td>OFF</td>
<td>3</td>
</tr>
<tr>
<td>Coordinates 2</td>
<td></td>
</tr>
<tr>
<td>SEQ</td>
<td>1</td>
</tr>
<tr>
<td>SIMUL</td>
<td>2</td>
</tr>
<tr>
<td>Grid 3</td>
<td></td>
</tr>
<tr>
<td>OFF</td>
<td>1</td>
</tr>
<tr>
<td>ON</td>
<td>2</td>
</tr>
<tr>
<td>Axes 4</td>
<td></td>
</tr>
<tr>
<td>In 3D Mode:</td>
<td></td>
</tr>
<tr>
<td>OFF</td>
<td>1</td>
</tr>
<tr>
<td>AXES</td>
<td>2</td>
</tr>
<tr>
<td>BOX</td>
<td>3</td>
</tr>
<tr>
<td>Not in 3D Mode:</td>
<td></td>
</tr>
<tr>
<td>OFF</td>
<td>1</td>
</tr>
<tr>
<td>ON</td>
<td>2</td>
</tr>
<tr>
<td>-----</td>
<td>---</td>
</tr>
<tr>
<td>Leading Cursor</td>
<td>5</td>
</tr>
<tr>
<td>OFF</td>
<td>1</td>
</tr>
<tr>
<td>ON</td>
<td>2</td>
</tr>
<tr>
<td>Labels</td>
<td>6</td>
</tr>
<tr>
<td>OFF</td>
<td>1</td>
</tr>
<tr>
<td>ON</td>
<td>1</td>
</tr>
<tr>
<td>Seq Axes</td>
<td>7</td>
</tr>
<tr>
<td>TIME</td>
<td>1</td>
</tr>
<tr>
<td>WEB</td>
<td>2</td>
</tr>
<tr>
<td>Custom</td>
<td>3</td>
</tr>
<tr>
<td>Solution Method</td>
<td>8</td>
</tr>
<tr>
<td>RK</td>
<td>1</td>
</tr>
<tr>
<td>EULER</td>
<td>2</td>
</tr>
<tr>
<td>Fields</td>
<td>9</td>
</tr>
<tr>
<td>SLPFLD</td>
<td>1</td>
</tr>
<tr>
<td>DIRFLD</td>
<td>2</td>
</tr>
<tr>
<td>FLDOFF</td>
<td>3</td>
</tr>
<tr>
<td>DE Axes</td>
<td>10</td>
</tr>
<tr>
<td>TIME</td>
<td>1</td>
</tr>
<tr>
<td>Y1-VS-Y2</td>
<td>2</td>
</tr>
<tr>
<td>----------------</td>
<td>---</td>
</tr>
<tr>
<td>T-VS-Y'</td>
<td>3</td>
</tr>
<tr>
<td>Y-VS-Y'</td>
<td>4</td>
</tr>
<tr>
<td>Y1-VS-Y2'</td>
<td>5</td>
</tr>
<tr>
<td>Y1'-VS-Y2'</td>
<td>6</td>
</tr>
<tr>
<td>XR Style</td>
<td>11</td>
</tr>
<tr>
<td>WIRE FRAME</td>
<td>1</td>
</tr>
<tr>
<td>HIDDEN SURFACE</td>
<td>2</td>
</tr>
<tr>
<td>CONTOUR LEVELS</td>
<td>3</td>
</tr>
<tr>
<td>WIRE AND CONTOUR</td>
<td>4</td>
</tr>
<tr>
<td>IMPLICIT PLOT</td>
<td>5</td>
</tr>
</tbody>
</table>
### setTable( )

<table>
<thead>
<tr>
<th>Parameter/Mode Setting</th>
<th>Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph &lt;-&gt; Table</td>
<td>1</td>
</tr>
<tr>
<td>OFF</td>
<td>1</td>
</tr>
<tr>
<td>ON</td>
<td>2</td>
</tr>
<tr>
<td>Independent</td>
<td>2</td>
</tr>
<tr>
<td>AUTO</td>
<td>1</td>
</tr>
<tr>
<td>ASK</td>
<td>2</td>
</tr>
<tr>
<td>Axes</td>
<td>4</td>
</tr>
</tbody>
</table>
Appendix D: Service and Warranty Information

Texas Instruments Support and Service

For general information

Home Page: education.ti.com
education.ti.com

KnowledgeBase and e-mail inquires: education.ti.com/support
education.ti.com/support

Phone: (800) TI-CARES; (800) 842-2737
For U.S., Canada, Mexico, Puerto Rico, and Virgin Islands only

International information: education.ti.com/international
education.ti.com/international

For technical support

KnowledgeBase and support by e-mail: education.ti.com/support
education.ti.com/support

Phone (not toll-free): (972) 917-8324
For product (hardware) service

Customers in the U.S., Canada, Mexico, Puerto Rico and Virgin Islands: Always contact Texas Instruments Customer Support before returning a product for service.

All other customers: Refer to the leaflet enclosed with this product (hardware) or contact your local Texas Instruments retailer/distributor.

Texas Instruments (TI) Warranty Information

Customers in the U.S. and Canada Only

One-Year Limited Warranty for Commercial Electronic Product

This Texas Instruments (“TI”) electronic product warranty extends only to the original purchaser and user of the product.

Warranty Duration. This TI electronic product is warranted to the original purchaser for a period of one (1) year from the original purchase date.

Warranty Coverage. This TI electronic product is warranted against defective materials and construction. THIS WARRANTY IS VOID IF THE PRODUCT HAS BEEN DAMAGED BY ACCIDENT OR UNREASONABLE USE, NEGLECT, IMPROPER SERVICE, OR OTHER CAUSES NOT ARISING OUT OF DEFECTS IN MATERIALS OR CONSTRUCTION.

Warranty Disclaimers. ANY IMPLIED WARRANTIES ARISING OUT OF THIS SALE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO THE ABOVE ONE-YEAR PERIOD. TEXAS INSTRUMENTS SHALL NOT BE LIABLE FOR LOSS OF USE OF THE PRODUCT OR OTHER INCIDENTAL OR CONSEQUENTIAL COSTS, EXPENSES, OR DAMAGES INCURRED BY THE CONSUMER OR ANY OTHER USER.

Some states/provinces do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusions may not apply to you.

Legal Remedies. This warranty gives you specific legal rights, and you may also have other rights that vary from state to state or province to province.
Warranty Performance. During the above one (1) year warranty period, your defective product will be either repaired or replaced with a reconditioned model of an equivalent quality (at TI's option) when the product is returned, postage prepaid, to Texas Instruments Service Facility. The warranty of the repaired or replacement unit will continue for the warranty of the original unit or six (6) months, whichever is longer. Other than the postage requirement, no charge will be made for such repair and/or replacement. TI strongly recommends that you insure the product for value prior to mailing.

Software. Software is licensed, not sold. TI and its licensors do not warrant that the software will be free from errors or meet your specific requirements. All software is provided “AS IS.”

Copyright. The software and any documentation supplied with this product are protected by copyright.
One-Year Limited Warranty for Commercial Electronic Product

This Texas Instruments electronic product warranty extends only to the original purchaser and user of the product.

**Warranty Duration.** This Texas Instruments electronic product is warranted to the original purchaser for a period of one (1) year from the original purchase date.

**Warranty Coverage.** This Texas Instruments electronic product is warranted against defective materials and construction. This warranty is void if the product has been damaged by accident or unreasonable use, neglect, improper service, or other causes not arising out of defects in materials or construction.

**Warranty Disclaimers.** Any implied warranties arising out of this sale, including but not limited to the implied warranties of merchantability and fitness for a particular purpose, are limited in duration to the above one-year period. Texas Instruments shall not be liable for loss of use of the product or other incidental or consequential costs, expenses, or damages incurred by the consumer or any other user.

Except as expressly provided in the One-Year Limited Warranty for this product, Texas Instruments does not promise that facilities for the repair of this product or parts for the repair of this product will be available.

Some jurisdictions do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusions may not apply to you.

**Legal Remedies.** This warranty gives you specific legal rights, and you may also have other rights that vary from jurisdiction to jurisdiction.

**Warranty Performance.** During the above one (1) year warranty period, your defective product will be either repaired or replaced with a new or reconditioned model of an equivalent quality (at TI’s option) when the product is returned to the original point of purchase. The repaired or replacement unit will continue for the warranty of the original unit or six (6) months, whichever is longer. Other than your cost to return the product, no charge will be made for such repair and/or replacement. TI strongly recommends that you insure the product for value if you mail it.

**Software.** Software is licensed, not sold. TI and its licensors do not warrant that the software will be free from errors or meet your specific requirements. **All software is provided “AS IS.”**

**Copyright.** The software and any documentation supplied with this product are protected by copyright.
All Other Customers

For information about the length and terms of the warranty, refer to your package and/or to the warranty statement enclosed with this product, or contact your local Texas Instruments retailer/distributor.

Battery Precautions

Take these precautions when replacing batteries:

- Do not leave batteries within the reach of children.
- Do not mix new and used batteries. Do not mix brands (or types within brands) of batteries.
- Do not mix rechargeable and non-rechargeable batteries.
- Install batteries according to polarity (+ and −) diagrams.
- Do not place non-rechargeable batteries in a battery recharger.
- Properly dispose of used batteries immediately.
- Do not incinerate or dismantle batteries.
TI-89 Titanium Shortcut Keys

**General**

- **[APPS]**
  - List of Flash applications
- **2nd [CE]**
  - Toggle between last two chosen applications or split screens
- **[2nd] [+] [+]**
  - Lighten or darken contrast
- **[ENTER]**
  - Calculate approximate answer
- **[2nd] [↑] [↓]**
  - Move cursor to top or bottom (in editors)
- **[2nd] [↑] [↓]**
  - Scroll tall objects in history
- **[2nd] [↑] [↓]**
  - Highlight left or right from cursor
- **[2nd] [↑] [↓]**
  - Page up or page down (in editors)
- **[2nd] [↑] [↓]**
  - Move cursor far left or far right

**On-screen Keyboard Map ( [ ] [EE] )**

Press **ESC** to exit the map.

The keyboard map displays shortcuts that are not marked on the keyboard. As shown below, press **[2nd]** and then the applicable key.

- **[G] [G]**
  - Access Greek letters (see next column)
- **[G] [G]**
  - (comment)
- **[G] [G]**
  - Copy graph coordinates to sysdata
- **[G] [G]**
  - ! (factorial)
- **[G] [G]**
  - Display FORMATS dialog box
- **[G] [1] [6]**
  - Run programs kbdprgm1( ) through kbdprgm6( )
- **[G] [STO]**
  - & (append)
- **[G] [ON]**
  - On-screen keyboard map
- **[G] [STO]**
  - @
- **[G] [ON]**
  - Turn off unit so that it returns to current application the next time you turn it on
- **[G] [2nd] [zero]**
  - ≤
- **[G] [2nd] [≥]**
  - ≥
- **[G] [ON]**
  - Copy graph coordinates to Home screen history

**Alpha Rules**

- **alpha**
  - Type one lowercase letter
- **1**
  - Type one uppercase letter
- **2nd [a-lock]**
  - Lowercase alpha lock
- **1 alpha**
  - Uppercase alpha lock
- **alpha**
  - Exit alpha lock

**3D Graphing**

- **[G] [G]**
  - Animate graph
- **[G] [G]**
  - Change animation speed
- **[G] [X], [G] [2nd]**
  - View along axis
- **[G] [2nd]**
  - Return to original view
- **[G] [1]**
  - Change graph format style
- **[G] [2nd]**
  - Expanded/normal view

**Greek Letters**

- **[G] [G]**
  - To access the Greek character set
- **[G] [2nd]**
  - + letter
  - To access lowercase Greek letters. Example:
    - \[ \alpha \] [W] displays \( \omega \)
- **[G] [2nd]**
  - + letter
  - To access uppercase Greek letters. Example:
    - \[ \alpha \] [W] displays \( \Omega \)

If you press a key combination that does not access a Greek letter, you get the normal letter for that key.
Voyage™ 200 Shortcut Keys

General

- **APPS**: List of Flash applications (if desktop is off)
- **2nd [ON]**: Toggle between last two chosen applications or split screens
- **D**: Copy graph coordinates to sysdata
- **F**: Display FORMATS dialog box
- **H**: Copy graph coordinates to Home screen history
- **N**: Create new variable
- **O**: Open existing variable
- **S**: Save copy as
- **. , +**: Lighten or darken contrast
- **ENTER**: Calculate approximate answer
- **ON**: Turn off unit so that it returns to current application the next time you turn it on
- **1 – 6**: Run programs kbdprgm1() through kbdprgm6()

On-screen Keyboard Map ([KEY])

Press [ESC] to exit the map.

![Keyboard Map Diagram]

See the table below for shortcuts that are not marked on the Voyage™ 200 keyboard. See the next column for accent marks and Greek letters.

- **2nd X**: © (comment)
- **X**: ≠
- **2nd O**: ≤
- **2nd I**: ≥

Editing

- Move cursor to top
- Move cursor to bottom
- Move cursor to far left
- Move cursor to far right
- Scroll tall objects in history
- Page up and page down
- Cut
- Copy
- Paste

3D Graphing

- Animates graph
- Change animation speed
- View along axis
- Return to original view
- Change graph format style
- Expanded/normal view

Accent Marks

- **A + letter**: à, è, ì, ò, ù, À, È, Ì, Ò, Ù
- **C + letter**: ç, Ç
- **E + letter**: á, é, í, ó, ú, ý, Á, É, Í, Ó, Ú, Ý
- **N + letter**: ã, ñ, õ, Æ, Ñ, Õ
- **O + letter**: â, õ, û, Â, Õ, Û
- **U + letter**: ä, ë, ï, ö, ü, ÿ, Ä, Ë, Ï, Ö, Ù, Ý
- **2nd G**: To access the Greek character set
- **2nd G + letter**: To access lowercase Greek letters. Example: **2nd G W** displays ω
- **2nd G [I] + letter**: To access uppercase Greek letters. Example: **2nd G [I] W** displays Ω

Greek Letters

- **2nd G**: To access the Greek character set
- **2nd G + letter**: To access lowercase Greek letters. Example: **2nd G W** displays ω
- **2nd G [I] + letter**: To access uppercase Greek letters. Example: **2nd G [I] W** displays Ω

If you press a key combination that does not access a Greek letter, you get the normal letter for that key.
Keystroke Differences

There are certain differences in keystrokes using the TI-89 Titanium / Voyage™ 200 for various operations. The following table shows the keystrokes for major commands for the two calculators.

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>TI-89 Titanium</th>
<th>Voyage 200</th>
</tr>
</thead>
<tbody>
<tr>
<td>LETTERS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>One lowercase letter (a-s, u, v, w)</td>
<td>[alpha] A-S, U-W</td>
<td>A-S, U-W</td>
</tr>
<tr>
<td>One lowercase letter (t, x, y, z)</td>
<td>T, X, Y, Z</td>
<td>T, X, Y, Z</td>
</tr>
<tr>
<td>Several lowercase letters</td>
<td>2nd [a-lock]</td>
<td></td>
</tr>
<tr>
<td>End several lowercase letters</td>
<td>[alpha]</td>
<td></td>
</tr>
<tr>
<td>Several uppercase letters</td>
<td>1 [a-lock]</td>
<td>2nd [CAPS]</td>
</tr>
<tr>
<td>End several uppercase letters</td>
<td>[alpha]</td>
<td>2nd [CAPS]</td>
</tr>
<tr>
<td>FUNCTION KEYS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F6</td>
<td>2nd [F6]</td>
<td>F6</td>
</tr>
<tr>
<td>F7</td>
<td>2nd [F7]</td>
<td>F7</td>
</tr>
<tr>
<td>F8</td>
<td>2nd [F8]</td>
<td>F8</td>
</tr>
<tr>
<td>NAVIGATION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scroll tall objects up or down in history</td>
<td>j, j</td>
<td>j, j</td>
</tr>
<tr>
<td>Move cursor far left or far right on entry line</td>
<td>2nd</td>
<td>2nd</td>
</tr>
<tr>
<td>Diagonal movement</td>
<td>and and and and</td>
<td>and and and and</td>
</tr>
<tr>
<td>FUNCTIONS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display Home screen</td>
<td>HOME</td>
<td>[CALC HOME]</td>
</tr>
<tr>
<td>Cut</td>
<td>[CUT]</td>
<td>X</td>
</tr>
<tr>
<td>Copy</td>
<td>[COPY]</td>
<td>C</td>
</tr>
<tr>
<td>Paste</td>
<td>[PASTE]</td>
<td>V</td>
</tr>
<tr>
<td>Catalog</td>
<td>CATALOG</td>
<td>2nd [CATALOG]</td>
</tr>
<tr>
<td>Display Units dialog box</td>
<td>2nd [UNITS]</td>
<td>[UNITS]</td>
</tr>
<tr>
<td>Sin</td>
<td>2nd [SIN]</td>
<td>SIN</td>
</tr>
<tr>
<td>Cos</td>
<td>2nd [COS]</td>
<td>COS</td>
</tr>
<tr>
<td>Tan</td>
<td>2nd [TAN]</td>
<td>TAN</td>
</tr>
<tr>
<td>LN</td>
<td>2nd [LN]</td>
<td>LN</td>
</tr>
<tr>
<td>$e^x$</td>
<td>[e^x]</td>
<td>2nd [e^x]</td>
</tr>
<tr>
<td>EE</td>
<td>EE</td>
<td>2nd EE</td>
</tr>
<tr>
<td>SYMBOLS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ (Conversion triangle)</td>
<td>2nd ▶</td>
<td>2nd ▶</td>
</tr>
<tr>
<td>_ (Underscore)</td>
<td>[ ]</td>
<td>2nd [ ]</td>
</tr>
<tr>
<td>Symbol</td>
<td>Voice Command</td>
<td>Action</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>--------</td>
</tr>
<tr>
<td>θ</td>
<td>Theta</td>
<td>973</td>
</tr>
<tr>
<td></td>
<td>(“With”)</td>
<td>2nd 1</td>
</tr>
<tr>
<td>'</td>
<td>Prime</td>
<td>2nd *</td>
</tr>
<tr>
<td>°</td>
<td>Degree</td>
<td>2nd *</td>
</tr>
<tr>
<td>∠</td>
<td>Angle</td>
<td>2nd ⟨</td>
</tr>
<tr>
<td>Σ</td>
<td>Sigma</td>
<td>CATALOG</td>
</tr>
<tr>
<td>x[⁻¹]</td>
<td>Reciprocal</td>
<td>CATALOG</td>
</tr>
<tr>
<td>Space</td>
<td></td>
<td>(alpha)[⁻¹]</td>
</tr>
</tbody>
</table>

**HIDDEN SHORTCUTS**

Place data in sysdata variable 0 D
Greek characters 0 1 G or 0 1 G
Keyboard map 0 Ε 0 [KEY]
Place data in Home screen history 0 0 H
Grave (à, ë, ì, ò, ù) 2nd [CHAR] 5 2nd A a, e, i, o, u
Cedilla (ç) 2nd [CHAR] 5 6 2nd C c
Acute (á, é, í, ó, ú, ý) 2nd [CHAR] 5 2nd E a, e, i, o, u, y
Tilde (ã, ŋ, ō) 2nd [CHAR] 5 6 2nd N a, n, o
Caret (â, ê, ì, õ, ũ) 2nd [CHAR] 5 2nd O a, e, i, o, u
Umlaut (ä, ë, í, ö, ü, ý) 2nd [CHAR] 5 2nd U a, e, i, o, u, y
? (Question mark) 2nd [CHAR] 3 2nd Q
β (Beta) 2nd [CHAR] 5 6 2nd S
# (Indirection) 2nd [CHAR] 3 2nd T
& (Append) 0 X (times) 2nd H
@ (Arbitrary) 0 STO® 2nd R
≠ (Not equal to symbol) 0 0 2nd V
! (Factorial) 0 0 2nd W
Comment (Circle-C) 0 [ ] 2nd X ©
New F1 3 0 N
Open F1 1 0 O
Save copy as F1 2 0 S
Format dialog box F1 1 0 F
Index

Symbols

!, factorial ........................................ 74, 907
➢, store ............................................ 596, 913
" , second notation ................................. 911
≠, /=, not equal .................................. 604, 906
#, indirection ....................................... 602, 909
√( ), square root .................................. 909
%, percent ........................................... 905
&, append ........................................... 602, 908
', minute notation ................................ 911
', prime ............................................. 911
Σ( ), sum ........................................... 266, 909
∫( ), integrate 80, 240, 242, 243, 247, 266, ..... 267, 908
* , multiply ........................................ 903
∫f(x)dx (graph math tool) ......................... 331, 334
+, add ............................................... 901
F1—F8 (function keys)
  moving among toolbar menus .................. 55
  selecting categories ........................... 28, 31
  selecting menus ................................ 49
  uses .............................................. 14
°, degree notation ................................ 764, 910, 911
- , negate ......................................... 905
−, subtract ......................................... 902
∠, angle ............................................ 910
.*, dot multiplication ............................ 905
.+ , dot addition .................................. 904
-. , dot subtraction ................................ 904
/, dot division ..................................... 905
^ , dot power ....................................... 905
/, divide ........................................... 903
<, less than ......................................... 604, 906
<<...>>, insufficient display memory .......... 229
= , equal ........................................... 604, 907
>, greater than ..................................... 604, 906
", comment ......................................... 581, 913
|, with ............................................. 80, 83, 239, 249, 913
∞, infinity .......................................... 277
← / [DEL] (delete character) ...................... 16
€ (hand modifier key)
  status ............................................ 38
2nd (second modifier key)
description ........................................ 13
status ................................................ 38
\textbf{2nd} [MEM] (MEMORY) ......................... 17
\textbf{2nd} [EE] (exponent key) ..................... 15
\textbf{2nd} [\textgreater] (measurement conversions) 16
\textbf{2nd} [RCL] (recall) ........................... 17
\textbf{2nd} [CATALOG] (Catalog)
\begin{itemize}
\item commands ..................................... 22
\item description ..................................... 21
\item exiting .......................................... 24
\item key command .................................... 17
\end{itemize}
\textbf{2nd} [CUSTOM] (Custom)
\begin{itemize}
\item description ..................................... 55
\item example .......................................... 56
\item key command .................................... 16
\end{itemize}
\textbf{2nd} [CHAR] (Character)
\begin{itemize}
\item description ..................................... 49
\item entering special characters ................. 10
\item key command .................................... 17
\item selecting characters ......................... 10
\end{itemize}
\textbf{2nd} [QUIT]
\begin{itemize}
\item calculator Home screen ..................... 24
\item entering commands ............................. 22
\item exiting the split-screen mode .............. 64
\item key command .................................... 17
\item turning off the calculator ................. 7
\end{itemize}
\begin{itemize}
\item \textgreater, convert ............................ 283, 912
\item Bin, display as binary ....................... 681, 790
\item Cylind, display as cylindrical vector ..... 804
\item DD, display as decimal angle ............. 807
\item Dec, display as decimal integer 681, 807
\item DMS, display as degree/minute/second 813
\item Grad, convert to Gradian angle measure 829
\item Hex, display as hexadecimal ............... 681, 830
\item Polar, display as polar vector .......... 855
\item Rad, convert to Radian angle measure .. 862
\item Rect, display as rectangular vector ...... 864
\item Sphere, display as spherical vector .... 882
\item (shift modifier key)
\begin{itemize}
\item description ..................................... 13
\item status ............................................ 38
\end{itemize}
\item (diamond modifier key)
\begin{itemize}
\item description ..................................... 13
\item status ............................................ 38
\end{itemize}
\item S (SAVE COPY AS)
\begin{itemize}
\item example .......................................... 54
\end{itemize}
\item STO\textgreater (store) key ..................... 17
\item (cursor keys)
\begin{itemize}
\item entering commands ............................. 23
\item opening Apps .................................. 28
\item selecting entry/answer pairs .............. 26
\item using the CHAR menu ....................... 10
\item -, negate ......................................... 156
\item \(\Pi\), product .................................. 266, 909
\item \(\theta_{\text{max}}\) window variable ............. 343
\item \(\theta_{\text{min}}\) window variable ............... 343
\item \(\theta_{\text{step}}\) window variable ............. 344
\item \(r\), radian ....................................... 910
\item T, transpose .................................... 886
\end{itemize}
Numerics
009AppB, page = 574 949
0b, binary indicator 914
0h, hexadecimal indicator 914
10^(), power of ten 912
Symbols
#, indirection 944
^, power 944
|, with 944
3D (three-dimensional) mode 38
3D graphing 375–409
animation 101, 391
CONTOUR LEVELS 104, 395
HIDDEN SURFACE 104, 395
WIRE AND CONTOUR 104, 395
WIRE FRAME 104, 395
ln() 838
logbase() 840
A
ABOUT screen 65
abs(), absolute value 767, 786
absolute value, abs() 767
accent marks
CHAR menu 17
accented characters 656
accuracy 941
add, + 901
Algebra menu 255, 258
algebra operations 782
All category 31
and (Boolean), and 249, 685, 786
and picture, AndPic 628
and, Boolean and 249, 605, 685
AndPic, and picture 787
Angle mode 187, 305, 923
angle mode 18, 38
status 38
angle(), angle 787
angle, (, angle 910
ans(), last answer 789
answer (last), ans() 220
APD (Automatic Power Down) 146
APD (Automatic Power Down) feature
during calculation or program 8
turning on after 8
append, & 602, 908
APPLICATIONS menu 181
APPLICATIONS menu (APPS) 50, 58
approx(), approximate 789
Approximate mode 166, 188, 203, 241
approximate, approx() 256
Apps (calculator software applications)
deleting 67
icon highlighted, last open 7
icons 5
names 28
opening 28, 58
Box Plot ................................................. 556
build
data, BldData .................................... 439, 596, 791
table, Table ......................................... 887
web, Build Web ....................................... 363
Build Web, build web ................................ 363, 365
build web, Build Web ................................. 365
BUSY ...................................................... 39
BUSY indicator ......................................... 204, 573
Busy/Pause status ..................................... 39

cables ................................................. 65, 67, 716, 733, 737
Calc menu ............................................ 265
calculator Home screen
  \[2\text{nd} \text{[QUIT]}\] ...................................... 17
changing entry/answer pairs ....................... 27
custom menu .......................................... 56
entering commands .................................. 22
function keys ........................................... 14
key command .......................................... 17
toolbar menus ........................................ 49
turning off the calculator ......................... 7
calculator software applications (Apps) ....... 7
  icons ...................................................... 5
Calculator-Based Laboratory system
  connecting .......................................... 68
Calculator-Based Laboratory. See CBL
Calculator-Based Ranger See CBR
Calculator-Based Ranger system
  connecting .......................................... 68
calculus operations .................................... 782
Catalog (\[2\text{nd} \text{[CATALOG]}\])
  commands ............................................ 22
description ............................................ 21
  exiting ................................................. 24
  key command ......................................... 17
  CATALOG menu ...................................... 191
categories
  All ...................................................... 31
  Apps desktop ....................................... 32
customizing ............................................ 33
  English ............................................... 31
  example of editing ................................ 34
  Graphing .............................................. 32
  Math ...................................................... 31
  Organizr (organizer) ............................... 32
  Science ............................................... 32
  selecting ............................................. 31
  selecting empty ..................................... 32
  SocialSt (social studies) ......................... 31
CBL
  programs ............................................ 633
  statistical data ..................................... 568, 569
CBL 2 system
  activity ............................................... 761
  connecting .......................................... 68
  programs .............................................. 761
CBL2
  get/return, Get ....................................... 825
  send list variable, Send .......................... 869
CBR
colNorm( ), matrix column norm 794
combinations, nCr( ) 846
comDenom( ), common denominator 795
command mark 661
command scripts 210, 661, 664
activity 752
commands
Flash Apps 21
Key 10, 12
comment, ♦ 581, 913
common denominator, comDenom( ) 256,
.......................... 257, 264
complex
factor, cFactor( ) 257, 773, 792
factor, cFactor(†) 938
mode, Complex Format 188, 923
modulus surface 402
numbers 73, 74
solve, cSolve( ) 240, 800
solve, cSolve(†) 938
tables 463
zeros, cZeros( ) 240, 257, 804
zeros, cZeros(†) 938
Complex Format mode 188, 923
complex format mode 18
Complex menu 257
conj( ), complex conjugate 795
connecting
TI ViewScreen overhead panel 68
TI-Presenter video adapter 68
Constant Memory 146
Constant Memory feature 17
constants 279
predefined 293
contact information 965
contour plots 396, 400, 401
DrwCtouir, draw contour 400
contour-level graphing 104, 395, 947
contrast
adjusting 5, 71
initial startup 5
contrast, adjusting 147
convert measurements 16
convert time, timeCnv( ) 890
convert, ▶ 283, 912
Coordinates graph format 315, 344
copy 212, 653
copy variable, CopyVar 596, 703
CopyVar, copy variable 703, 796
cos( ), cosine 796
cos⁻¹( ), arccosine 797
cosh( ), hyperbolic cosine 798
cosh⁻¹( ), hyperbolic arccosine 798
cot( ), cotangent 798
cot⁻¹( ), inverse cotangent 799
coth( ), hyperbolic cotangent 799
coth⁻¹( ), inverse hyperbolic cotangent 799
crossP( ), cross product 799
csc( ), cosecant 799
csc⁻¹( ), inverse cosecant 800
csch( ), hyperbolic cosecant 800
<table>
<thead>
<tr>
<th>Term</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>csch⁻¹( ), inverse hyperbolic cosecant</td>
<td>800</td>
</tr>
<tr>
<td>cSolve( ), complex solve</td>
<td>240, 800</td>
</tr>
<tr>
<td>cSolve(†), complex solve</td>
<td>938</td>
</tr>
<tr>
<td>cubic regression, CubicReg</td>
<td>546, 945</td>
</tr>
<tr>
<td>CubicReg, cubic regression</td>
<td>546, 802, 945</td>
</tr>
<tr>
<td>cumSum( ), cumulative sum</td>
<td>534, 803</td>
</tr>
<tr>
<td>Current folder mode</td>
<td>187, 922</td>
</tr>
<tr>
<td>Current folder status</td>
<td>38</td>
</tr>
<tr>
<td>Current mode</td>
<td>18</td>
</tr>
<tr>
<td>cursor</td>
<td></td>
</tr>
<tr>
<td>3D graph</td>
<td>382</td>
</tr>
<tr>
<td>deleting characters</td>
<td>16</td>
</tr>
<tr>
<td>free-moving</td>
<td>319, 345, 352, 361, 381, 419</td>
</tr>
<tr>
<td>hidden surface</td>
<td>384</td>
</tr>
<tr>
<td>in the history area</td>
<td>26</td>
</tr>
<tr>
<td>location following APD</td>
<td>8</td>
</tr>
<tr>
<td>moving</td>
<td>172</td>
</tr>
<tr>
<td>off the curve</td>
<td>385</td>
</tr>
<tr>
<td>Selecting a command</td>
<td>23</td>
</tr>
<tr>
<td>trace</td>
<td>321</td>
</tr>
<tr>
<td>Viewing entries</td>
<td>26</td>
</tr>
<tr>
<td>cursor keys (↑↓←→)</td>
<td></td>
</tr>
<tr>
<td>entering commands</td>
<td>23</td>
</tr>
<tr>
<td>opening Apps</td>
<td>28</td>
</tr>
<tr>
<td>selecting entry/answer pairs</td>
<td>26</td>
</tr>
<tr>
<td>using the CHAR menu</td>
<td>10</td>
</tr>
<tr>
<td>CustmOff, custom toolbar off</td>
<td>230, 803</td>
</tr>
<tr>
<td>CustmOn, custom toolbar on</td>
<td>230, 803</td>
</tr>
<tr>
<td>CUSTOM (2nd [CUSTOM]) menu</td>
<td>56</td>
</tr>
<tr>
<td>description</td>
<td>55</td>
</tr>
<tr>
<td>key command</td>
<td>16</td>
</tr>
<tr>
<td>CUSTOM axes (sequence)</td>
<td>363</td>
</tr>
<tr>
<td>CUSTOM custom plots</td>
<td>357, 434, 435</td>
</tr>
<tr>
<td>CUSTOM menu</td>
<td>230</td>
</tr>
<tr>
<td>custom plots, CUSTOM</td>
<td>357, 434, 435</td>
</tr>
<tr>
<td>custom toolbar</td>
<td></td>
</tr>
<tr>
<td>Custom Units mode</td>
<td>189, 926</td>
</tr>
<tr>
<td>Custom, define toolbar</td>
<td>621, 806</td>
</tr>
<tr>
<td>customer support and service</td>
<td>965</td>
</tr>
<tr>
<td>cut</td>
<td>212, 653</td>
</tr>
<tr>
<td>cycle picture, CyclePic</td>
<td>500, 629, 804</td>
</tr>
<tr>
<td>Cycle, cycle</td>
<td>804</td>
</tr>
<tr>
<td>CyclePic, cycle picture</td>
<td>629, 804</td>
</tr>
<tr>
<td>cylindrical vector display, Cylind</td>
<td>804</td>
</tr>
<tr>
<td>cZeros( ), complex zeros</td>
<td>240, 257, 804</td>
</tr>
<tr>
<td>cZeros(†), complex zeros</td>
<td>938</td>
</tr>
</tbody>
</table>

**D**

<table>
<thead>
<tr>
<th>Term</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>d( ), first derivative</td>
<td>79, 247, 265, 267, 806</td>
</tr>
<tr>
<td>darker/lighter</td>
<td>147</td>
</tr>
<tr>
<td>data (new), NewData</td>
<td>846</td>
</tr>
<tr>
<td>data filtering</td>
<td>757</td>
</tr>
<tr>
<td>data plots</td>
<td>119</td>
</tr>
<tr>
<td>Data/Matrix Editor</td>
<td></td>
</tr>
<tr>
<td>cell width</td>
<td>527</td>
</tr>
<tr>
<td>column header</td>
<td>528, 531, 532, 533</td>
</tr>
<tr>
<td>filling</td>
<td>525</td>
</tr>
<tr>
<td>list variable</td>
<td>520, 522</td>
</tr>
<tr>
<td>new, NewData</td>
<td>597</td>
</tr>
<tr>
<td>scrolling</td>
<td>525</td>
</tr>
</tbody>
</table>
shift, shift( ) .......................... 875
sorting columns ......................... 535
statistical plots .......................... 551
values .................................. 523, 524
data/matrix editor ....................... 469
data•mat( ) ................................. 806
date
reset .................................... 48
setting .................................. 40
dayOfWk( ), day of week ............ 807
DE (differential equation) mode ..... 38
decimal
angle display, ▶DD ...................... 807
integer display, ▶Dec .................. 681, 807
define toolbar, Toolbar .............. 621, 892
Define, define .......................... 357, 413, 446, 742, 808
define, Define .......................... 224, 271, 309, 350, 357,
378, 413, 446, 472, 479, 591, 596,
627, .................................. 742, 808
DEG (degree) mode ...................... 38
degree notation, ° ..................... 764, 910, 911
degree/minute/second display, ▶DMS 813
delete character (.currentTarget / [DEL]) ..... 16
deleting
folder, DelFold ......................... 596, 808
variable, DelVar 238, 270, 596, 600,
809
variables of type ...................... 705
deleting variables ..................... 143
DelFold, delete folder ................. 596, 808
DelType .................................. 809
DelType command ...................... 705
DelVar, delete variable ............... 596, 600, 809
denominator ........................... 795
derivatives ................................ 79
first derivative, d( ) ................. 806
first derivative, d( ) 79, 247, 265, 267
numeric derivative, nDeriv( ) 266, 846
Derivatives (graph math tool) .... 331, 334, 346, 353, 809
desolve( ), solution ................. 266, 444, 809
det( ), matrix determinant .......... 810
diag( ), matrix diagonal ............. 811
dialog box
CLOCK .................................. 41
edit categories .......................... 33
menu indicator ........................ 53
MODE .................................... 17
to open Apps .......................... 28
dialog box, define, Dialog .......... 621, 811
dialog boxes ............................ 179
Dialog, define dialog box .......... 621, 811
Diamond modifier key (.currentTarget)
description ........................... 13
status .................................. 38
differential equations
DIRFLD, direction field .............. 415, 423
first order ............................. 425, 444
FLDOFF, field off .................... 415, 424, 451
diagram .................................. 410-452
initial conditions .................... 421
second order .......................... 427, 444
<table>
<thead>
<tr>
<th>Command</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>polar, DrawPol</td>
<td>487, 632, 814</td>
</tr>
<tr>
<td>slope, DrawSlp</td>
<td>495, 631, 814</td>
</tr>
<tr>
<td>tangent line, LineTan</td>
<td>632, 837</td>
</tr>
<tr>
<td>vertical line, LineVert</td>
<td>632, 837</td>
</tr>
<tr>
<td>DrawInv, draw inverse</td>
<td>488, 632, 813</td>
</tr>
<tr>
<td>DrawParm, draw parametric</td>
<td>487, 632, 814</td>
</tr>
<tr>
<td>DrawPol, draw polar</td>
<td>487, 632, 814</td>
</tr>
<tr>
<td>DrawSlp, draw slope</td>
<td>495, 631, 814</td>
</tr>
<tr>
<td>drop-down menu, DropDown</td>
<td>622</td>
</tr>
<tr>
<td>DropDown, drop-down menu</td>
<td>622, 814</td>
</tr>
<tr>
<td>DrwCtour, draw contour</td>
<td>400, 632, 815</td>
</tr>
<tr>
<td>end, else</td>
<td>831</td>
</tr>
<tr>
<td>else, ElseIf</td>
<td>831</td>
</tr>
<tr>
<td>Else, else</td>
<td>607</td>
</tr>
<tr>
<td>else, Else</td>
<td>607</td>
</tr>
<tr>
<td>Elseif, else if</td>
<td>607, 816</td>
</tr>
<tr>
<td>else if, ElseIf</td>
<td>478, 607</td>
</tr>
<tr>
<td>for, EndFor</td>
<td>582, 611, 823</td>
</tr>
<tr>
<td>function, EndFunc</td>
<td>587, 824</td>
</tr>
<tr>
<td>if, EndIf</td>
<td>582, 606, 831</td>
</tr>
<tr>
<td>loop, EndLoop</td>
<td>614, 841</td>
</tr>
<tr>
<td>program, EndPrgm</td>
<td>128, 591, 856</td>
</tr>
<tr>
<td>toolbar, EndTBar</td>
<td>621, 892</td>
</tr>
<tr>
<td>try, EndTry</td>
<td>636, 892</td>
</tr>
<tr>
<td>while, EndWhile</td>
<td>613, 895</td>
</tr>
<tr>
<td>EndCustm, end custom</td>
<td>621, 803</td>
</tr>
<tr>
<td>EndDlog, end dialog</td>
<td>621, 811</td>
</tr>
<tr>
<td>EndFor, end for</td>
<td>582, 611, 823</td>
</tr>
<tr>
<td>EndFunc, end function</td>
<td>587, 824</td>
</tr>
<tr>
<td>EndIf, end if</td>
<td>582, 606, 831</td>
</tr>
<tr>
<td>EndLoop, end loop</td>
<td>614, 841</td>
</tr>
<tr>
<td>EndPrgm, end program</td>
<td>128, 591, 856</td>
</tr>
<tr>
<td>EndTBar, end toolbar</td>
<td>621, 892</td>
</tr>
<tr>
<td>EndTry, end try</td>
<td>636, 892</td>
</tr>
<tr>
<td>EndWhile, end while</td>
<td>613, 895</td>
</tr>
<tr>
<td>English category</td>
<td>31</td>
</tr>
<tr>
<td>entry line</td>
<td></td>
</tr>
<tr>
<td>clearing the history area</td>
<td>28</td>
</tr>
<tr>
<td>cursor rests on</td>
<td>26</td>
</tr>
<tr>
<td>inserting commands</td>
<td>22</td>
</tr>
<tr>
<td>recalling</td>
<td>26</td>
</tr>
<tr>
<td>entry( ), entry</td>
<td>817</td>
</tr>
<tr>
<td>entry, entry( )</td>
<td>219</td>
</tr>
<tr>
<td>entry/answer pairs</td>
<td>27</td>
</tr>
<tr>
<td>status</td>
<td>39</td>
</tr>
<tr>
<td>EOS (Equation Operating System)</td>
<td>943</td>
</tr>
<tr>
<td>equal, =</td>
<td>604, 906</td>
</tr>
<tr>
<td>Equation Operating System (EOS)</td>
<td>943</td>
</tr>
<tr>
<td>equations, solving</td>
<td>666, 670, 674, 675</td>
</tr>
<tr>
<td>error conditions after APD</td>
<td>8</td>
</tr>
<tr>
<td>errors and troubleshooting</td>
<td>952</td>
</tr>
<tr>
<td>Topic</td>
<td>Page(s)</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Circular definition</td>
<td>595</td>
</tr>
<tr>
<td>clear error, ClrErr</td>
<td>636, 793</td>
</tr>
<tr>
<td>Memory error</td>
<td>713, 714</td>
</tr>
<tr>
<td>Out-of-memory</td>
<td>273</td>
</tr>
<tr>
<td>pass error, PassErr</td>
<td>636, 854</td>
</tr>
<tr>
<td>programs</td>
<td>635</td>
</tr>
<tr>
<td>transmission</td>
<td>724, 735</td>
</tr>
<tr>
<td>warnings</td>
<td>922</td>
</tr>
<tr>
<td>Estep window variable</td>
<td>418</td>
</tr>
<tr>
<td>Euler method</td>
<td>414, 439</td>
</tr>
<tr>
<td>evaluate polynomial, polyEval( )</td>
<td>855</td>
</tr>
<tr>
<td>EXACT mode status</td>
<td>38</td>
</tr>
<tr>
<td>exact( ), exact</td>
<td>817</td>
</tr>
<tr>
<td>Exact/Approx mode</td>
<td>166, 188, 203, 240, 241, 242, 925</td>
</tr>
<tr>
<td>exact/approx mode</td>
<td>18</td>
</tr>
<tr>
<td>example</td>
<td></td>
</tr>
<tr>
<td>changing mode settings</td>
<td>19</td>
</tr>
<tr>
<td>editing categories</td>
<td>34</td>
</tr>
<tr>
<td>restoring the default custom menu</td>
<td>56</td>
</tr>
<tr>
<td>selecting menu options</td>
<td>51</td>
</tr>
<tr>
<td>turning off the clock</td>
<td>47</td>
</tr>
<tr>
<td>turning on/off the custom menu</td>
<td>56</td>
</tr>
<tr>
<td>using dialog boxes</td>
<td>54</td>
</tr>
<tr>
<td>using the CHAR menu</td>
<td>11</td>
</tr>
<tr>
<td>using the keyboard map</td>
<td>12, 13</td>
</tr>
<tr>
<td>examples, previews, activities</td>
<td></td>
</tr>
<tr>
<td>3D graphing</td>
<td>101, 750</td>
</tr>
<tr>
<td>additional graphing topics</td>
<td>110</td>
</tr>
<tr>
<td>angle modes</td>
<td>81</td>
</tr>
<tr>
<td>baseball</td>
<td>763</td>
</tr>
<tr>
<td>CBL 2 program</td>
<td>761</td>
</tr>
<tr>
<td>complex factors</td>
<td>773</td>
</tr>
<tr>
<td>complex modulus surface</td>
<td>402</td>
</tr>
<tr>
<td>complex numbers</td>
<td>73</td>
</tr>
<tr>
<td>complex zeroes</td>
<td>766</td>
</tr>
<tr>
<td>constants and measurement units</td>
<td>84</td>
</tr>
<tr>
<td>converging web plots</td>
<td>366</td>
</tr>
<tr>
<td>cos(x)=sin(x) activity</td>
<td>748</td>
</tr>
<tr>
<td>cubic polynomial</td>
<td>766</td>
</tr>
<tr>
<td>data filtering</td>
<td>757</td>
</tr>
<tr>
<td>data/matrix editor</td>
<td>117</td>
</tr>
<tr>
<td>decomposing a rational function</td>
<td>755</td>
</tr>
<tr>
<td>derivatives</td>
<td>79</td>
</tr>
<tr>
<td>detecting discontinuities</td>
<td>92</td>
</tr>
<tr>
<td>differential equations</td>
<td>105</td>
</tr>
<tr>
<td>diverging web plots</td>
<td>368</td>
</tr>
<tr>
<td>expanding expressions</td>
<td>76</td>
</tr>
<tr>
<td>factorial</td>
<td>73</td>
</tr>
<tr>
<td>factoring polynomials</td>
<td>259</td>
</tr>
<tr>
<td>Fibonacci sequence</td>
<td>373</td>
</tr>
<tr>
<td>finding roots</td>
<td>75</td>
</tr>
<tr>
<td>function graphing</td>
<td>86, 89</td>
</tr>
<tr>
<td>graphing functions</td>
<td>87</td>
</tr>
<tr>
<td>implicit derivatives</td>
<td>79</td>
</tr>
<tr>
<td>implicit plots</td>
<td>407</td>
</tr>
<tr>
<td>integrals</td>
<td>80</td>
</tr>
<tr>
<td>log to any base</td>
<td>81</td>
</tr>
<tr>
<td>memory management</td>
<td>138</td>
</tr>
<tr>
<td>number bases</td>
<td>136</td>
</tr>
<tr>
<td>numeric solver</td>
<td>133</td>
</tr>
<tr>
<td>oscillating web plots</td>
<td>369</td>
</tr>
</tbody>
</table>
parametric graphing 94, 764
path of a ball 94
polar rose 96
pole-corner problem 741
population 119
predator-prey model 371, 435
prime factors 73
programming 128, 130, 637
Pythagorean theorem 741
quadratic formula 743
rational factors 773
real factors 773
reducing expressions 76
sampling 775
second-order differential equation 427, 444
sequence graphing 98
solving inequalities 78
solving linear equations 77, 260
split screen 114, 764
standard annuity 769
statistics 119
symbolic manipulation 82
tables 112
text operations 131
third-order differential equation 431
time value of money 771
trees and forest 98
tutorial script with the text editor 752
variable management 138
vectors 80
examples, previews, activities
log to any base 81
exclusive or (Boolean), xor 605, 685, 895
exclusive or picture, XorPic 629, 896
Exec, execute assembly language 644, 818
execute assembly language, Exec 644, 818
execute program, Prgm 128, 591, 856
Exit, exit 818
exp(list(), expression to list) 818
expand(), expand 76, 745, 766, 818
expand, expand() 76, 256, 259, 745, 766
exponent key (2nd [EE]) 15
exponent, e 15, 815
Exponential Format mode 170, 188, 923
exponential Format mode 18
exponential regression, ExpReg 546, 820, 945
expr(), string to expression 601, 602, 620, 819
ExpReg, exponential regression 546, 820, 945
expressions 24, 159, 160, 161, 172
expanding 76
expression to list, exp(list()) 818
reducing 76
string to expression, expr() 601, 602, 620, 819
Extract menu 257
<table>
<thead>
<tr>
<th>Term</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>eyeφ z-axis window variable</td>
<td>378, 387, 388</td>
</tr>
<tr>
<td>eyeθ x-axis window variable</td>
<td>378, 387, 388</td>
</tr>
<tr>
<td>eyeψ rotation window variable</td>
<td>378, 387, 389</td>
</tr>
</tbody>
</table>

**F**

<table>
<thead>
<tr>
<th>Term</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>factor( ), factor</td>
<td>75, 745, 773, 820</td>
</tr>
<tr>
<td>factor, factor( )</td>
<td>75, 240, 256, 259, 745, 773</td>
</tr>
<tr>
<td>factorial, !</td>
<td>74, 907</td>
</tr>
<tr>
<td>factoring</td>
<td>259</td>
</tr>
<tr>
<td>activity</td>
<td>773</td>
</tr>
<tr>
<td>false message</td>
<td>275</td>
</tr>
<tr>
<td>family of curves</td>
<td>479</td>
</tr>
<tr>
<td>FCC statement</td>
<td>2</td>
</tr>
<tr>
<td>Fibonacci sequence</td>
<td>373</td>
</tr>
<tr>
<td>field off, FLDOFF</td>
<td>415, 424, 451</td>
</tr>
<tr>
<td>field picture, fldpic</td>
<td>419</td>
</tr>
<tr>
<td>Fill, matrix fill</td>
<td>821</td>
</tr>
<tr>
<td>Flash applications</td>
<td>181, 195, 273, 689, 690,</td>
</tr>
<tr>
<td>deleting</td>
<td>724</td>
</tr>
<tr>
<td>FLASH APPLICATIONS (_APPS)</td>
<td></td>
</tr>
<tr>
<td>accessing Apps not listed</td>
<td>59</td>
</tr>
<tr>
<td>description</td>
<td>50</td>
</tr>
<tr>
<td>key command</td>
<td>16</td>
</tr>
<tr>
<td>Flash, upgrading operating system</td>
<td>731, 732,</td>
</tr>
<tr>
<td>FLDOFF, field off</td>
<td>415, 424, 451</td>
</tr>
<tr>
<td>fldpic, field picture</td>
<td>419</td>
</tr>
<tr>
<td>fldres window variable</td>
<td>418</td>
</tr>
<tr>
<td>floor( ), floor</td>
<td>749, 821</td>
</tr>
<tr>
<td>floor, floor( )</td>
<td>749</td>
</tr>
<tr>
<td>fMax( ), function maximum</td>
<td>822</td>
</tr>
<tr>
<td>fMin( ), function minimum</td>
<td>822</td>
</tr>
<tr>
<td>FnOff, function off</td>
<td>627, 822</td>
</tr>
<tr>
<td>FnOn, function on</td>
<td>627, 823</td>
</tr>
<tr>
<td>folders</td>
<td>187, 922</td>
</tr>
<tr>
<td>delete, DelFold</td>
<td>596, 808</td>
</tr>
<tr>
<td>get/return, getFold( )</td>
<td>826</td>
</tr>
<tr>
<td>locking/unlocking</td>
<td>703</td>
</tr>
<tr>
<td>new, NewFold</td>
<td>597, 847</td>
</tr>
<tr>
<td>pasting name</td>
<td>705, 706</td>
</tr>
<tr>
<td>renaming</td>
<td>700, 703</td>
</tr>
<tr>
<td>setting, setFold( )</td>
<td>617, 871</td>
</tr>
<tr>
<td>transmitting</td>
<td>719, 720, 721, 722, 723</td>
</tr>
<tr>
<td>VARLINK</td>
<td>693, 694, 695, 696, 698, 699,</td>
</tr>
<tr>
<td>For, for</td>
<td>823</td>
</tr>
<tr>
<td>for, For</td>
<td>582, 611, 823</td>
</tr>
<tr>
<td>format string, format( )</td>
<td>602, 620, 627</td>
</tr>
<tr>
<td>format( ), format string</td>
<td>823</td>
</tr>
<tr>
<td>FORMATS dialog box</td>
<td>104, 105, 315, 393,</td>
</tr>
<tr>
<td>fpart( ), function part</td>
<td>824</td>
</tr>
<tr>
<td>fractions</td>
<td>256, 264, 755, 857</td>
</tr>
<tr>
<td>free-moving cursor</td>
<td>319, 345, 352, 361, 381,</td>
</tr>
<tr>
<td>Frobenius norm, norm( )</td>
<td>849</td>
</tr>
<tr>
<td>full-screen mode</td>
<td></td>
</tr>
</tbody>
</table>
2nd [QUIT] ... 17
Apps desktop ... 37
changing from split-screen ... 64
displaying Apps in ... 64
FUNC (function) mode ... 38
Func, program function ... 587, 824
function keys ([F1]-[F8])
   moving among toolbar menus ... 55
   selecting categories ... 28, 31
   selecting menus ... 49
functions ... 21, 159
delayed simplification ... 247
graphing ... 302
maximum, fMax( ) ... 240, 266, 822
minimum, fMin( ) ... 240, 266, 822
multistatement ... 477
off, FnOff ... 311, 627, 822
on, FnOn ... 311, 627, 823
part, fPart( ) ... 824
program function, Func ... 587, 824
user-defined 196, 223, 271, 378, 475,
   ... 477, 585, 587, 808

G
Garbage collection message ... 709, 710, 711,
   ... 712, 713
gcd( ), greatest common divisor ... 824
get time zone, GetTmZn( ) ... 828
Get, get/return CBL/CBR value ... 568, 634
Get, get/return CBL2/CBR value ... 825
get/return calculator, GetCalc ... 634, 727, 728, 825
CBL/CBR value, Get ... 568, 634
CBL2/CBR value, Get ... 825
denominator, getDenom( ) ... 257, 826
folder, getFold( ) ... 596, 617, 826
key, getKey( ) ... 619, 826
key, getKey(†) ... 929, 931
mode, getMode( ) ... 617, 827
number, getNum( ) ... 257, 827
type, getType( ) ... 237, 596, 828
units, getUnits( ) ... 617, 829
GetCalc, get/return calculator ... 634, 727, 728,
   ... 825
getConfig( ), get/return configuration ... 617, 825
date, getDate( ) ... 825
get/return denumerator ... 826
date format, getDtFmt( ) ... 826
date string, getDtStr( ) ... 826
folder, getFold( ) ... 596, 617
key, getKey( ) ... 619, 826
key, getKey(†) ... 929, 931
mode, getMode( ) ... 617, 827
number, getNum( ) ... 257, 827
type, getType( ) ... 237, 596, 828
units, getUnits( ) ... 617, 829
getTime( ), get time ... 827
date format, getTmFmt( ) ... 827
date string, getTmStr( ) ... 827
time zone, getTmZn( ) ... 828
type, getType( ) ... 596, 828
<table>
<thead>
<tr>
<th>Term</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>getUnits() / get/return units</td>
<td>617, 829</td>
</tr>
<tr>
<td>global variables</td>
<td>600</td>
</tr>
<tr>
<td>go to, Goto</td>
<td>593, 609, 616</td>
</tr>
<tr>
<td>Goto, go to</td>
<td>829</td>
</tr>
<tr>
<td>GRAD (gradian) mode</td>
<td>38</td>
</tr>
<tr>
<td>GRAD(gradian) mode</td>
<td>81</td>
</tr>
<tr>
<td>Gradian angle mode</td>
<td>81</td>
</tr>
<tr>
<td>gradian, G</td>
<td>910</td>
</tr>
<tr>
<td>graph</td>
<td>18, 38</td>
</tr>
<tr>
<td>number mode</td>
<td>38</td>
</tr>
<tr>
<td>Graph 2 mode</td>
<td>188, 925</td>
</tr>
<tr>
<td>Graph mode</td>
<td>187, 203, 304, 342, 349, 356, 377, 412, 922</td>
</tr>
<tr>
<td>graph mode status</td>
<td>38</td>
</tr>
<tr>
<td>graph number mode status</td>
<td>38</td>
</tr>
<tr>
<td>Graph Order graph format</td>
<td>316, 414</td>
</tr>
<tr>
<td>Graph, graph</td>
<td>830</td>
</tr>
<tr>
<td>graph, Graph</td>
<td>309, 473, 480, 628</td>
</tr>
<tr>
<td>Graph&lt;&gt;Table, table-graph</td>
<td>456</td>
</tr>
<tr>
<td>graphing category</td>
<td>32</td>
</tr>
<tr>
<td>graphs</td>
<td>62, 64</td>
</tr>
<tr>
<td>graphs and graphing</td>
<td></td>
</tr>
<tr>
<td>$\int f(x)dx$</td>
<td>331, 334</td>
</tr>
<tr>
<td>3D</td>
<td>375-409</td>
</tr>
<tr>
<td>animation</td>
<td>500</td>
</tr>
<tr>
<td>Arc</td>
<td>331, 337, 346, 353</td>
</tr>
<tr>
<td>clearing, ClrGraph</td>
<td>475, 627, 677, 794</td>
</tr>
<tr>
<td>contour plots</td>
<td>396, 400, 401</td>
</tr>
<tr>
<td>coordinates</td>
<td>87, 319</td>
</tr>
<tr>
<td>custom axes</td>
<td>363</td>
</tr>
<tr>
<td>custom plots</td>
<td>357, 434, 435</td>
</tr>
<tr>
<td>Derivatives</td>
<td>331, 334, 346, 353</td>
</tr>
<tr>
<td>differential equations</td>
<td>410-453</td>
</tr>
<tr>
<td>Distance</td>
<td>331, 335, 346, 353</td>
</tr>
<tr>
<td>drawing</td>
<td>489, 491, 492, 493, 494, 495, 496, 629</td>
</tr>
<tr>
<td>family of curves</td>
<td>479</td>
</tr>
<tr>
<td>formats</td>
<td>315, 344, 414</td>
</tr>
<tr>
<td>functions</td>
<td>302</td>
</tr>
<tr>
<td>functions off, FnOff</td>
<td>627, 822</td>
</tr>
<tr>
<td>functions on, FnOn</td>
<td>627, 823</td>
</tr>
<tr>
<td>graph databases</td>
<td>503</td>
</tr>
<tr>
<td>graph, Graph</td>
<td>473, 628, 830</td>
</tr>
<tr>
<td>Home screen</td>
<td>471, 473</td>
</tr>
<tr>
<td>implicit plots</td>
<td>404, 407</td>
</tr>
<tr>
<td>independent variable</td>
<td>471</td>
</tr>
<tr>
<td>Inflection</td>
<td>331, 335</td>
</tr>
<tr>
<td>Intersection</td>
<td>331, 333</td>
</tr>
<tr>
<td>inverse functions</td>
<td>488</td>
</tr>
<tr>
<td>line styles</td>
<td>311, 343, 351, 358, 378</td>
</tr>
<tr>
<td>math functions</td>
<td>331</td>
</tr>
<tr>
<td>matrix data</td>
<td>469</td>
</tr>
<tr>
<td>Maximum</td>
<td>331, 333</td>
</tr>
<tr>
<td>Minimum</td>
<td>87, 331, 333</td>
</tr>
<tr>
<td>modes</td>
<td>187, 203, 304, 342, 349, 356, 377, 412, 922</td>
</tr>
<tr>
<td>native independent variable</td>
<td>471</td>
</tr>
<tr>
<td>nested functions</td>
<td>476</td>
</tr>
<tr>
<td>operations</td>
<td>782</td>
</tr>
</tbody>
</table>
overview . . . 302, 340, 347, 354, 375, 410
panning ........................................ 323
parametric ....................................... 347
pausing ........................................... 318
pictures .......................................... 497, 499
piecewise functions ......................... 475
pixels ............................................. 941
polar ............................................. 340
programs ......................................... 627
QuickCenter ................................. 324
recall graph database, RclGDB . 629, 863
selecting functions . . . . . 309, 350, 358, 413
sequence ................................. 354–374
setting, setGraph( ) . 617, 628, 871
Shade ................................. 332, 337
shading, Shade ......................... 632, 875
simultaneous graphs ..................... 481
split screen ......................... 482, 485, 506
store graph database, StoGDB . 629, 884
style, Style ................................. 628, 885
Tangent ......................................... 331, 336, 346, 353
text ............................................. 496
time plots .................................. 357, 363, 434, 435
trace, Trace 321, 628, 751, 761, 763,
................................................. 766, 892
tracing 87, 321, 324, 346, 353, 362,
381, ........................................... 420
two-graph mode . 482, 483, 484, 506
Value . 331, 332, 346, 353, 362, 382,
420
viewing window . 313, 343, 351, 359,
379
web plots .................................. 357, 363, 364
window variables 313, 343, 351, 359,
................................................. 379
Y= editor 86, 89, 305, 343, 349, 356,
377, ........................................... 412, 471
Zero ........................................... 331, 333
zoom . 325, 345, 353, 362, 381, 628
zoom factors .................................. 326, 328
zoom Memory .................................. 326, 329
greater than or equal, ≥, >= ........ 604, 907
greater than, > ......................... 604, 907
greatest common divisor, gcd( ) .... 824
Greek characters ................................ 10, 657, 658
Grid graph format ......................... 316

H
Hand modifier key (⌜) ........................................... 38
status ............................................. 38
Hardware version ......................... 234
hexadecimal
   display, ▶Hex ......................... 681, 830
   indicator, 0h ......................... 914
hidden surface .......................... 104, 384, 395
highlighting
   characters when editing ........ 13
to view full name of App ........ 6
| highlighting text .................. 652 |
| --- | --- |
| Histogram 557 |
| History area status 39 |
| history area 207, 208, 664 |
| History indicator 27 |
| Home icon 24 |
| Home screen 205 |
| Home screen. See calculator home screen |
| hyperbolic arccosine \( \cosh^{-1}( ) \) 798 |
| arcsine, \( \sinh^{-1}( ) \) 878 |
| arctangent, \( \tanh^{-1}( ) \) 889 |
| cosecant, \( \text{csch}( ) \) 800 |
| cosine, \( \cosh( ) \) 798 |
| cotangent, \( \coth( ) \) 799 |
| secant, \( \text{sech}( ) \) 869 |
| sine, \( \sinh( ) \) 878 |
| tangent, \( \tanh( ) \) 888 |
| implied multiplication ... 160, 350 |
| Independent AUTO/ASK, independent auto/ask 459, 464 |
| independent auto/ask, Independent AUTO/ASK 457, 459, 464 |
| indirection, \# 602, 909, 944 |
| inequalities 78 |
| infinity, \( \infty \) 277 |
| Inflection (graph math tool) 331, 335 |
| initial conditions 421 |
| initial startup 5 |
| input string, InputSt 601, 619, 728, 832 |
| Input, input 832 |
| input, Input 619, 628 |
| InputSt, input string 601, 619, 728, 832 |
| insert mode \([2\text{nd}][\text{INS}]\) 16 |
| inString( ), within string 602, 833 |
| instructions 159 |
| calculator Home screen 24 |
| Catalog 21 |
| insufficient display memory, \(<<...>>\) 229 |
| int( ), integer 833 |
| intDiv( ), integer divide 683, 833 |
| integer divide, intDiv( ) 683 |
| integer part, iPart( ) 99, 833 |
| integer, int( ) 833 |
| integrate, \( \int( ) \) 80, 240, 242, 243, 247, 266, 267, 908 |
| international/accented characters 10 |
| Intersection (graph math tool) 331, 333 |
| inverse cosecant, \( \csc^{-1}( ) \) 800 |
inverse cotangent, \( \cot^{-1}(x) \) .......................... 799
inverse hyperbolic
cosecant, \( \operatorname{csch}^{-1}(x) \) ..................... 800
cotangent, \( \coth^{-1}(x) \) .......................... 799
secant, \( \operatorname{sech}^{-1}(x) \) .......................... 869
inverse, \( x^{-1} \) ........................................... 912
iPart( ), integer part ..................................... 833
isArchiv() ...................................................... 833
isArchiv(), is archived ................................... 694
isClkOn( ), is clock on .................................... 833
isLocked ....................................................... 834
isLocked(), is locked ....................................... 694
isPrime( ), prime test ..................................... 834
isVar( ) .......................................................... 834
isVAR(), is variable ........................................ 694
Item, menu item ............................................. 622, 624, 834

K
key commands
keyboard map ................................................. 12
special characters .......................................... 10
keyboard ......................................................... 148
\( \text{2nd} \) (second) key ................................... 150
\( \text{shift} \) (shift) key ....................................... 150
\( \diamondsuit \) (diamond) key ................................ 150
\( \text{alpha} \) (alpha) key ................................... 150
key codes ....................................................... 619
map ............................................................. 10, 11, 656, 657
shortcuts ....................................................... 656, 657
keys
function ......................................................... 14

L
label, Lbl ...................................................... 593, 609, 616
Labels graph format ......................................... 316
Language mode ................................................. 189, 926
language mode
changing mode setting ...................................... 19
viewing ......................................................... 18
last answer .................................................. 153, 164, 217, 220
last entry ..................................................... 153, 217, 218
Lbl, label ......................................................... 834
lcm, least common multiple ................................ 835
Leading Cursor graph format .............................. 316
least common multiple, lcm ............................... 835
left( ), left ..................................................... 835
left, left( ) ..................................................... 257, 602
left-right split screen
setting .......................................................... 59
setting initial Apps .......................................... 61
status ........................................................... 36
less than or equal, \( \neq \) .................................. 907
less than or equal, \( \leq, \preceq \) ......................... 604
less than, \( < \) ................................................. 604, 906
lighter/darker ................................................ 147
limit( ), limit .................................................. 835
limit, limit( ) .................................................. 247, 266, 268
Line, draw line ............................................... 631, 836
linear regression, LinReg ................................. 547, 837, 945
LineHorz, draw horizontal line ......................... 632, 836
LineTan, draw tangent line ............................... 632, 837
<table>
<thead>
<tr>
<th>LineVert, draw vertical line</th>
<th>632, 837</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link transmission table</td>
<td>740</td>
</tr>
<tr>
<td>linking and transmitting</td>
<td>869, 870</td>
</tr>
<tr>
<td>calculator to calculator</td>
<td>633, 716, 718, 719, 721, 727, 728, 730, 731</td>
</tr>
<tr>
<td>cancelling</td>
<td>724</td>
</tr>
<tr>
<td>errors</td>
<td>724, 734, 735</td>
</tr>
<tr>
<td>Flash applications</td>
<td>719, 720, 721, 726, 727, 727, 727</td>
</tr>
<tr>
<td>folders</td>
<td>719, 720, 722, 723, 724</td>
</tr>
<tr>
<td>get/return CBL/CBR value</td>
<td>568, 634</td>
</tr>
<tr>
<td>get/return CBL2/CBR value</td>
<td>825</td>
</tr>
<tr>
<td>program</td>
<td>633, 727, 728</td>
</tr>
<tr>
<td>send chat</td>
<td>727, 728</td>
</tr>
<tr>
<td>send list variable</td>
<td>635, 869</td>
</tr>
<tr>
<td>send to calculator</td>
<td>634, 727, 728</td>
</tr>
<tr>
<td>variables</td>
<td>719, 720, 721, 722, 723</td>
</tr>
<tr>
<td>LinReg, linear regression</td>
<td>547, 837, 945</td>
</tr>
<tr>
<td>list difference, Δlist( )</td>
<td>838</td>
</tr>
<tr>
<td>list to matrix, list\text{\textbackslash mat}( )</td>
<td>838</td>
</tr>
<tr>
<td>list\text{\textbackslash mat}( ), list to matrix</td>
<td>532, 838</td>
</tr>
<tr>
<td>lists</td>
<td></td>
</tr>
<tr>
<td>augment/concatenate</td>
<td>790</td>
</tr>
<tr>
<td>cross product, crossP( )</td>
<td>799</td>
</tr>
<tr>
<td>cumulative sum, cumSum( )</td>
<td>803</td>
</tr>
<tr>
<td>difference, Δlist( )</td>
<td>838</td>
</tr>
<tr>
<td>dot product, dotP( )</td>
<td>813</td>
</tr>
<tr>
<td>expression to list</td>
<td>818</td>
</tr>
<tr>
<td>list to matrix, list\text{\textbackslash mat}( )</td>
<td>838</td>
</tr>
<tr>
<td>matrix to list, mat\text{\textbackslash list}( )</td>
<td>843</td>
</tr>
<tr>
<td>maximum, max( )</td>
<td>843</td>
</tr>
<tr>
<td>mid-string, mid( )</td>
<td>844</td>
</tr>
<tr>
<td>minimum, min( )</td>
<td>845</td>
</tr>
<tr>
<td>new data, NewData</td>
<td>597, 846</td>
</tr>
<tr>
<td>new, newList( )</td>
<td>847</td>
</tr>
<tr>
<td>operations</td>
<td>782</td>
</tr>
<tr>
<td>product, product( )</td>
<td>856</td>
</tr>
<tr>
<td>sort ascending, SortA</td>
<td>882</td>
</tr>
<tr>
<td>sort descending, SortD</td>
<td>882</td>
</tr>
<tr>
<td>summation, sum( )</td>
<td>866, 885</td>
</tr>
<tr>
<td>table variables</td>
<td>466</td>
</tr>
<tr>
<td>variables</td>
<td>516</td>
</tr>
<tr>
<td>ln( ), natural logarithm</td>
<td>838</td>
</tr>
<tr>
<td>LnReg, logarithmic regression</td>
<td>547, 839, 946</td>
</tr>
<tr>
<td>local variable, Local</td>
<td>589, 594, 596, 597</td>
</tr>
<tr>
<td>Local, local variable</td>
<td>839</td>
</tr>
<tr>
<td>lock variable, Lock</td>
<td>597</td>
</tr>
<tr>
<td>Lock, lock variable</td>
<td>839</td>
</tr>
<tr>
<td>locked/archived variable status</td>
<td>39</td>
</tr>
<tr>
<td>log to any base</td>
<td>81</td>
</tr>
<tr>
<td>log( ), logarithm</td>
<td>840</td>
</tr>
<tr>
<td>logarithm, log( )</td>
<td>840</td>
</tr>
<tr>
<td>logarithmic regression, LnReg</td>
<td>547, 839, 946</td>
</tr>
<tr>
<td>logarithms</td>
<td>838, 840</td>
</tr>
<tr>
<td>logistic regression, Logistic</td>
<td>547, 946</td>
</tr>
<tr>
<td>Logistic, logistic regression</td>
<td>547, 841, 946</td>
</tr>
<tr>
<td>Loop, loop</td>
<td>841</td>
</tr>
</tbody>
</table>
loop, Loop ........................................ 614
LU, matrix lower-upper decomposition .... 842

M
matdata( ) .......................................... 842
matlist( ), matrix to list ................. 843
math category ..................................... 31
MATH menu ........................................ 331
MATH menu ([2nd][MATH]) ............... 49
math operations ................................. 24, 783
matrices
  augment/concenate, augment( ) ........ 747,
  column dimension, colDim( ) ....... 794
  column norm, colNorm( ) .......... 794
  copying ........................................ 537
  cumulative sum, cumSum( ) ....... 803
data from a graph ............................. 469
determinant, det( ) ......................... 810
diagonal, diag( ) ............................ 811
dimension, dim( ) ............................ 811
dot addition, + ............................... 904
dot division, ./ ............................... 905
dot multiplication, ( ...................... 905
dot power, ^ ............................... 905
dot subtraction, - ............................ 904
eigenvalue, eigVl( ) ......................... 816
eigenvector, eigVc( ) ......................... 816
filling, Fill ..................................... 821
identity, identity( ) ......................... 831
list to matrix, listmat( ) ................. 838
locking ........................................... 530
lower-upper decomposition, LU ....... 842
matrix to list, matlist( ) ................. 843
maximum, max( ) ............................ 843
minimum, min( ) ............................ 845
new data, NewData ......................... 597, 846
new, newMat( ) ............................... 847
operations ....................................... 783
pretty print ....................................... 519
product, product( ) ......................... 856
QR factorization, QR ......................... 860
random, randMat( ) ......................... 747, 863
reduced row echelon form, rref( ) .... 262,
row addition, rowAdd( ) ................ 867
row dimension, rowDim( ) ............... 867
row echelon form, ref( ) ................. 864
row multiplication and addition,
mRowAdd( ) .................................. 845
row norm, rowNorm( ) ..................... 867
row operation, mRow( ) ................. 845
row swap, rowSwap( ) ...................... 868
submatrix, subMat( ) ....................... 885
summation, sum( ) ......................... 866, 885
transpose, T ................................. 886
matrix to list, matlist( ) ................. 843
max( ), maximum ............................ 843
Maximum (graph math tool) .............. 331, 333
mean( ), mean ................................. 843
measurement
conversions ([2nd [↑]]) .......................... 16
median( ), median .......................... 843
medium-medium line regression, MedMed
 ........................................... 547, 844, 946
MedMed, medium-medium line regression
 ........................................... 547, 844, 946
memory
 archiving, Archive 596, 708, 709, 789
 checking .............................. 689, 690
 insufficient display memory, <<...>> .
 229
 resetting .............................. 689, 690
 unarchive, Unarchiv .597, 708, 709,
 893
 VARLINK screen 691, 693, 694, 695,
 .............................. 696, 698, 699, 700, 701, 708
 MEMORY ([2nd [MEM]]) .......................... 17
 Memory (zoom) .............................. 326, 329
 Memory error .............................. 713
 menu item, Item ........................... 622, 624, 834
 Menus
 APPLICATIONS ([APPS]) .......................... 50, 58
 menus .............................. 175
 Algebra .............................. 255, 258
 APPLICATIONS .............................. 181
 Calc .............................. 265
canceling .............................. 55
 CATALOG .............................. 191
 CHAR .............................. 10, 17, 49
 Clean Up .............................. 189
 Complex .............................. 257
 CUSTOM .............................. 230
 custom .............................. 623, 626
 CUSTOM ([2nd [CUSTOM])] .......................... 16, 55, 56
 Extract .............................. 257
 FLASH APPLICATIONS ([APPS]) .......................... 50, 59
 MATH .............................. 331
 options .............................. 13
 selecting options .............................. 50
 submenu options .............................. 52
toolbar .............................. 175, 230
 Trig .............................. 257
 using .............................. 175
 messages
 BATT .............................. 204, 949, 952
 false .............................. 275
 Garbage collection .............................. 709, 710, 711,
 .............................. 712, ..................... 713
 insufficient display memory, <<...>> .
 229
 true .............................. 275
 undef (undefined) .............................. 278
 messages See also errors and
troubleshooting
 mid( ), mid-string .............................. 844
 mid-string, mid( ) .............................. 602
 min( ), minimum .............................. 845
 Minimum (graph math tool) .............................. 87, 331, 333
 minute notation, ' .............................. 911
 mod( ), modulo .............................. 845
 modes .............................. 185
3D (three-dimensional) ................. 38
Angle ........................................ 18, 38, 187, 305, 923
APPROX ...................................... 38
Approximate ................................. 166, 188, 203, 925
Apps desktop ............................... 18
AUTO .......................................... 38
Auto ............................................ 166, 188, 203, 242, 925
Base ............................................. 18, 188, 926
Complex Format .................. 18, 188, 923
complex format ............................ 18
current ........................................ 18
Current folder ............................. 187, 922
Custom Units ............................... 926
custom units .................................. 18
DE (differential equation) .............. 38
DEG (degree) ................................. 38
Display Digits ...................... 169, 187, 922
display digits .................................. 18
EXACT ......................................... 38
Exact/Approx ................................. 166, 188, 203, 240, 241, 242, 925
exact/approx ................................. 18
Exponential Format .................. 170, 188, 923
exponential format ....................... 18
fullscreen .................................. 17, 31, 37, 61, 64
FUNC (function) ............................. 38
get/return, getMode( ) .................. 617, 827
GRAD (gradian) ............................... 38
Graph ........................................... 187, 203, 304, 342, 349, 356, 377, 412
graph .......................................... 18
Graph 2 ....................................... 188, 925
graph number ............................... 38
graph type .................................... 38
grayed out .................................... 18
insert ([2nd] [INS]) ...................... 16
Language ..................................... 189, 926
language ...................................... 18, 19
Number of Graphs ..................... 188, 925
overwrite ([2nd] [INS]) ............... 16
PAR (parametric) ............................ 38
POL (polar) .................................... 38
Pretty Print ................................. 18, 166, 188, 924
RAD (radian) ................................. 38
SEQ (sequence) ............................... 38
setting in programs .................. 616
setting, setMode( ) .............. 617, 628, 872
settings ........................................ 17
Split App .................................... 188, 925
Split Screen ................................. 188, 924
split screen ................................. 6, 18, 31, 36, 38, 59, 61, 62, 64
Unit System ................................. 189, 926
unit system .................................... 18
Vector Format ............................... 188, 924
vector format ............................... 18
modifier keys ([2nd] ♠ ↑ ☯) ............ 13
status ............................................ 38
move variable, MoveVar ............ 597
MoveVar, move variable ............ 845
mRow( ), matrix row operation ...... 845
mRowAdd( ), matrix row multiplication and addition ........................................... 845
multiply, * .............................................................................................................. 903
multistatement functions ......................................................................................... 477

N
natural log base, e ......................................................................................................... 277
natural logarithm, \ln( ) .............................................................................................. 838
ncontour window variable ......................................................................................... 379
nCr( ), combinations ................................................................................................. 846
ncurves window variable ............................................................................................ 417
nDeriv( ), numeric derivative .................................................................................. 266
negate, - ......................................................................................................................... 156
negation key (\([-\)])) .................................................................................................. 14
negative numbers .......................................................................................................... 14
new
  data, NewData ............................................................................................................ 569
  folder, NewFold .......................................................................................................... 597
  list, newList( ) ........................................................................................................... 847
  matrix, newMat( ) ..................................................................................................... 847
  picture, NewPic ......................................................................................................... 597
  plot, NewPlot ............................................................................................................ 556
  problem, NewProb .................................................................................................... 190
NewData, new data ...................................................................................................... 518
NewFold, new folder ..................................................................................................... 597
newList( ), new list ...................................................................................................... 847
newMat( ), new matrix ............................................................................................... 847
NewPic, new picture ..................................................................................................... 597
NewPlot, new plot ........................................................................................................ 556
NewProb, new problem ............................................................................................... 190

nInt( ), numeric integral ............................................................................................ 266
nmax window variable ................................................................................................. 359
nmin window variable ................................................................................................. 359
norm( ), Frobenius norm ........................................................................................... 849
not (Boolean), not ........................................................................................................ 684
not equal, \neq, \neq .................................................. 604
not, Boolean not ........................................................................................................... 605
nPr( ), permutations ..................................................................................................... 849
nSolve( ), numeric solution ....................................................................................... 256
number bases ................................................................................................................ 680
  Boolean operations ..................................................................................................... 684
  conversions .................................................................................................................... 681
  math operations .......................................................................................................... 682
Number of Graphs mode .............................................................................................. 188
numbers
  irrational ...................................................................................................................... 240
  negative ....................................................................................................................... 156
  rational ......................................................................................................................... 240
numeric
derivative, nDeriv( ) ................................................................................................... 266
integral, nInt( ) ............................................................................................................ 266
solution, nSolve( ) ....................................................................................................... 256
numeric keypad ............................................................................................................. 14
numeric solver ............................................................................................................ 666
  equations ..................................................................................................................... 666
  graphing ....................................................................................................................... 675
  split screens ................................................................................................................ 676
  variables ....................................................................................................................... 670

O
on/off ............................................................................................................................ 145
OneVar, one-variable statistic ........................................ 850
operating system .................................................. 733, 734, 735
operating system (OS)
  downloading .......................................................... 69
Operating System (OS) version ................................... 234
operating system, upgrading ....................................... 731, 732, 733
operators .................................................................. 159
or (Boolean), or .......................................................... 685
or, Boolean or ............................................................ 605, 685, 851
ord( ), numeric character code .................................. 602, 851
Organizr (organizer) category ..................................... 32
OS ........................................................................... 731, 732, 733
OS (Operating System) version .................................. 234
Out-of-memory error .................................................. 273
Output, output ............................................................ 851
output, Output .............................................................. 620, 627
overwrite mode ([2nd] [INS]) ...................................... 16

P
P>Rx( ), rectangular x coordinate ................................ 852
P>Ry( ), rectangular y coordinate ................................ 852
panning .................................................................... 323
PAR (parametric) mode .............................................. 38
parallelepiped activity ............................................... 750
parametric graphing ................................................... 347
parentheses, brackets, and braces ................................ 160, 943
part( ), part .............................................................. 852
pass error, PassErr .................................................... 636
PassErr, pass error ...................................................... 854
paste ...................................................................... 212, 653
PAUSE ...................................................................... 39
PAUSE indicator ........................................................ 204
Pause, pause .............................................................. 854
pause, Pause .............................................................. 620, 636
percent, % ................................................................. 905
permutations, nPr( ) .................................................. 849
pictures .................................................................. 497, 499
  and, AndPic ............................................................ 628, 787
cycle, CyclePic ........................................................ 629, 804
deleting ................................................................... 500
exclusive or, XorPic .................................................. 629, 896
new, NewPic ............................................................. 597, 629, 847
recall, RclPic .............................................................. 629, 863
replace, RplcPic ........................................................ 629, 868
storing, StoPic ........................................................... 629, 884
piecewise functions .................................................. 475
pixel
  change, PxlChg ......................................................... 631, 858
circle, PxlCrcl ............................................................ 631, 858
horizontal line, PxlHorz ............................................. 632, 858
line, PxlLine ................................................................ 496, 631, 859
off, PxlOff ................................................................ 631, 859
on, PxlOn ................................................................ 496, 631, 859
test, pxlTest( ) ............................................................ 631, 859
text, PxlText .............................................................. 631, 859
vertical line, PxlVert ................................................... 632, 860
plots
  clearing ................................................................ 554
data ..................................................................... 119
new, NewPlot ............................................................. 556, 628, 848
<table>
<thead>
<tr>
<th>Term</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>off, PlotsOff</td>
<td>311, 628, 855</td>
</tr>
<tr>
<td>on, PlotsOn</td>
<td>311, 628, 855</td>
</tr>
<tr>
<td>selecting</td>
<td>553, 560</td>
</tr>
<tr>
<td>tracing</td>
<td>562</td>
</tr>
<tr>
<td>viewing window</td>
<td>561</td>
</tr>
<tr>
<td>Y= Editor</td>
<td>559</td>
</tr>
<tr>
<td>PlotsOff, plots off</td>
<td>311, 855</td>
</tr>
<tr>
<td>PlotsOn, plots on</td>
<td>311, 855</td>
</tr>
<tr>
<td>plotStep window variable</td>
<td>359</td>
</tr>
<tr>
<td>plotStrt window variable</td>
<td>359</td>
</tr>
<tr>
<td>point</td>
<td></td>
</tr>
<tr>
<td>change, PtChg</td>
<td>631, 857</td>
</tr>
<tr>
<td>off, PtOff</td>
<td>631, 857</td>
</tr>
<tr>
<td>on, PtOn</td>
<td>631, 858</td>
</tr>
<tr>
<td>test, ptTest( )</td>
<td>631, 858</td>
</tr>
<tr>
<td>text, PtText</td>
<td>631, 858</td>
</tr>
<tr>
<td>POL (polar) mode</td>
<td>38</td>
</tr>
<tr>
<td>polar</td>
<td></td>
</tr>
<tr>
<td>coordinate, R►Pθ( )</td>
<td>862</td>
</tr>
<tr>
<td>coordinate, R►Pr( )</td>
<td>862</td>
</tr>
<tr>
<td>graphing</td>
<td>340</td>
</tr>
<tr>
<td>vector display, ▶Polar</td>
<td>855</td>
</tr>
<tr>
<td>polyEval( ), evaluate polynomial</td>
<td>855</td>
</tr>
<tr>
<td>polynomials</td>
<td>259, 267</td>
</tr>
<tr>
<td>activity</td>
<td>766</td>
</tr>
<tr>
<td>evaluate, polyEval( )</td>
<td>855</td>
</tr>
<tr>
<td>random, randPoly( )</td>
<td>863</td>
</tr>
<tr>
<td>popup menu, PopUp</td>
<td>619</td>
</tr>
<tr>
<td>PopUp, popup menu</td>
<td>619, 856</td>
</tr>
<tr>
<td>power of ten, 10^( )</td>
<td>912</td>
</tr>
<tr>
<td>power regression, PowerReg</td>
<td>547, 856, 946</td>
</tr>
<tr>
<td>power, ^</td>
<td>904, 944</td>
</tr>
<tr>
<td>PowerReg, power regression</td>
<td>547, 856, 946</td>
</tr>
<tr>
<td>pretty print</td>
<td>87, 166, 206</td>
</tr>
<tr>
<td>Pretty Print mode</td>
<td>18, 166, 188, 924</td>
</tr>
<tr>
<td>Prgm, execute program</td>
<td>128, 591, 856</td>
</tr>
<tr>
<td>prime number test, isPrime( )</td>
<td>834</td>
</tr>
<tr>
<td>prime numbers</td>
<td>75</td>
</tr>
<tr>
<td>prime, '</td>
<td>911</td>
</tr>
<tr>
<td>problems (new), NewProb</td>
<td>190, 848</td>
</tr>
<tr>
<td>product ID</td>
<td>233</td>
</tr>
<tr>
<td>Product ID (identifier)</td>
<td>234</td>
</tr>
<tr>
<td>product( ), product</td>
<td>856</td>
</tr>
<tr>
<td>product, Π( )</td>
<td>266, 909</td>
</tr>
<tr>
<td>Program Editor</td>
<td>29</td>
</tr>
<tr>
<td>programs and programming</td>
<td>21, 572–644</td>
</tr>
<tr>
<td>arguments</td>
<td>584</td>
</tr>
<tr>
<td>assembly language</td>
<td>641, 644</td>
</tr>
<tr>
<td>branching</td>
<td>582, 606, 609</td>
</tr>
<tr>
<td>calling another program</td>
<td>591</td>
</tr>
<tr>
<td>CBL</td>
<td>633</td>
</tr>
<tr>
<td>CBL 2 system</td>
<td>761</td>
</tr>
<tr>
<td>CBR</td>
<td>633</td>
</tr>
<tr>
<td>CBR system</td>
<td>761</td>
</tr>
<tr>
<td>clear error, ClrErr</td>
<td>636, 793</td>
</tr>
<tr>
<td>clear graph, ClrGraph</td>
<td>475, 627, 794</td>
</tr>
<tr>
<td>clear home, ClrHome</td>
<td>794</td>
</tr>
<tr>
<td>clear I/O, ClrIO</td>
<td>575, 620, 794</td>
</tr>
<tr>
<td>clear table, ClrTable</td>
<td>794</td>
</tr>
</tbody>
</table>
menus ................................ 622, 626
multicmd lines .......................... 580
operations .............................. 784
output ................................ 574, 582, 620
output, Output .......................... 620, 627, 851
pass error, PassErr ...................... 636, 851
passing values .......................... 584
pause, Pause ........................... 620, 636, 854
popup menu, PopUp ...................... 619, 854
prompt, Prompt( ) ........................ 619, 856
request, Request ....................... 620, 622, 865
return, Return .......................... 589, 592, 865
running ................................ 572
stop, Stop ................................ 579, 884
stopping ................................ 573
subroutines ............................. 591
tables ................................ 627
text, Text ................................ 621, 622, 890
Then, Then ................................ 606, 607, 831
title, Title ............................... 622, 890
try, Try ................................ 636, 892
variables ................................ 593
while, While .............................. 613, 895
Prompt( ), prompt ......................... 619, 857
prompt, Prompt( ) ........................ 619
propFrac, proper fraction .................. 256, 264, 755
PtChg, point change ...................... 631, 857
PtOff, point off .......................... 631, 857
PtOn, point on ............................ 631, 858
ptTest( ), point test ..................... 631, 858
PtText, point text ....................... 631, 858
PxlChg, pixel change ..................... 631, 858
PxlCrcl, pixel circle ...................... 631, 858
PxlHorz, pixel horizontal line ........... 632, 858
PxlLine, pixel line ....................... 496, 631, 858
PxlOff, pixel off ........................ 496, 631, 859
PxlOn, pixel on .......................... 496, 631, 859
pxlTest( ), pixel test .................... 631, 859
PxlText, pixel text ...................... 631, 859
PxlVert, pixel vertical line .............. 632, 860

Q
QR factorization, QR .................... 860
QR, QR factorization ..................... 860
quadratic regression, QuadReg 547, 861,
QuadReg, quadratic regression 547, 861,
quartic regression, QuartReg 548, 861,
QuartReg, quartic regression 548, 861,
QuickCenter ............................. 324
Quit ([2nd] [QUIT]) ....................... 17

R
r, radian ................................ 910
R>Pθ( ), polar coordinate ............... 862
R>Pr( ), polar coordinate ............... 862
RAD (radian) mode ...................... 38
radian, \( r \) ........................................ 910
rand( ), random number ..................... 862
randMat( ), random matrix ................... 747, 863
randNorm( ), random norm ................... 863
random
  matrix, randMat( ) .......................... 747, 863
  norm, randNorm( ) .......................... 863
number seed, RandSeed ........................ 747, 863
number, rand( ) .............................. 862
polynomial, randPoly( ) ........................ 863
randPoly( ), random polynomial ............. 863
RandSeed, random number seed ............... 747, 863
rational functions activity .................... 755
RclGDB, recall graph database ................ 505, 629, 863
RclPic, recall picture .......................... 629, 863
real( ), real .................................. 863
recall
graph database, RclGDB ....................... 505, 629, 863
  picture, RclPic .............................. 629, 863
Recall ([2nd] [RCL]) ........................... 17
reciprocal, \( x^{-1} \) .......................... 912
rectangular x coordinate, \( Px( ) \) ........... 852
rectangular y coordinate, \( Py( ) \) ........... 852
rectangular-vector display, \( \text{Rect} \) ....... 864
reduced row echelon form, \( \text{rref}( ) \) ...... 262, 747
ref( ), row echelon form ...................... 864
regressions .................................. 837
  cubic, CubicReg ............................. 546, 802, 945
  exponential, ExpReg ....................... 546, 820, 945
  formulas ................................... 945, 947
  linear regression, LinReg .................. 547, 837, 945
  logarithmic, LnReg ......................... 547, 839, 946
  logistic, Logistic .......................... 547, 841, 946
  medium-medium line, MedMed ............... 547, 844,
  power regression, PowerReg ................ 547, 856,
  quadratic formula activity .................. 743
  quadratic, QuadReg ......................... 547, 861, 946
  quartic, QuartReg ........................... 548, 861, 946
  selecting .................................. 546
  sinusoidal, SinReg .......................... 548, 879, 946
remain( ), remainder ......................... 865
Rename, rename ................................ 865
rename, Rename ................................ 597
replace picture, RplcPic ...................... 629, 868
Request, request ............................. 865
request, Request ............................. 620, 622
reserved names ................................ 943
results ..................................... 24
return See get/return ......................... 865
Return, return ................................ 478, 589, 592
return, Return ................................ 478, 589, 592
right( ), right ................................ 866
right, right( ) ................................ 257, 603
root ......................................... 866
roots ........................................ 75
rotate( ), rotate .......................... 686, 866
rotate, rotate( ) .......................... 603, 686
round( ), round .......................... 867
row echelon form, ref( ) .................. 864
rowAdd( ), matrix row addition .......... 867
rowDim( ), matrix row dimension ......... 867
rowNorm( ), matrix row norm .......... 867
rowSwap( ), matrix row swap .......... 868
RplcPic, replace picture .................. 629, 868
rref( ), reduced row echelon form .... 262, 747
Runge-Kutta method ......................... 414, 435, 948

S
sampling activity .......................... 775
SAVE COPY AS (S)
   example .................................. 54
Scatter plots ............................... 555
scientific notation ........................ 15, 157
scripts ..................................... 210, 661, 664
   activity .................................. 752
   tutorial .................................. 752
scrolling ..................................... 26, 228, 463
sec( ), secant ................................ 868
sec^{-1}( ), inverse secant ................. 869
secant, sec( ) ................................ 868
sech( ), hyperbolic secant ............... 869
sech^{-1}( ), inverse hyperbolic secant .. 869
Second modifier key (2nd)
   description ............................... 13
   status ..................................... 38
second notation, .......................... 911
selecting categories ....................... 31
send chat, SendChat ......................... 727, 728, 870
send list variable, Send ................. 635, 860
send to calculator, SendCalc ............. 634, 727, 728
   Send, send list variable .......... 635, 869
SendCalc, send to calculator .......... 634, 727, 728
   SendChat, send chat ............ 727, 728, 870
SEQ (sequence) mode ......................... 38
seq( ), sequence .......................... 870
sequence graphing .......................... 354–374
serial number ............................... 233
set
   folder, setFold( ) ......................... 617, 871
   graph, setGraph( ) ....................... 617, 628
   mode, setMode( ) ......................... 617, 628
   table, setTable( ) ......................... 459, 617
   units, setUnits( ) ......................... 617
Set factors (zoom) ......................... 326, 328
setDate( ), set date ........................ 870
setDtFmt( ), set date format .............. 870
setFold( ), set folder ....................... 617, 871
setGraph( ), set graph ..................... 617, 628, 871
setMode( ), set mode ....................... 617, 628, 872
setTable( ), set table ....................... 459, 617, 873
setTime( ), set time ........................ 873
setTmFmt( ), set time format .............. 873
setTmZn( ), set time zone .................. 874
setUnits( ), set units ....................... 617, 874
<table>
<thead>
<tr>
<th>Concept/Command</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shade (graph math tool)</td>
<td>332, 337</td>
</tr>
<tr>
<td>Shade, shade</td>
<td>632, 875</td>
</tr>
<tr>
<td>shade, Shade</td>
<td>632</td>
</tr>
<tr>
<td>Shift modifier key (↑)</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>13</td>
</tr>
<tr>
<td>status</td>
<td>38</td>
</tr>
<tr>
<td>shift( ), shift</td>
<td>533, 603, 687, 875</td>
</tr>
<tr>
<td>shift, shift( )</td>
<td>603, 687</td>
</tr>
<tr>
<td>show statistical results, ShowStat</td>
<td>548, 876</td>
</tr>
<tr>
<td>ShowStat, show statistical results</td>
<td>548, 876</td>
</tr>
<tr>
<td>sign( ), sign</td>
<td>876</td>
</tr>
<tr>
<td>simplification</td>
<td></td>
</tr>
<tr>
<td>delayed</td>
<td>247</td>
</tr>
<tr>
<td>rules</td>
<td>244</td>
</tr>
<tr>
<td>stopping</td>
<td>246</td>
</tr>
<tr>
<td>simult( ), simultaneous equations</td>
<td>877</td>
</tr>
<tr>
<td>simultaneous equations, simult( )</td>
<td>262</td>
</tr>
<tr>
<td>sin( ), sine</td>
<td>877</td>
</tr>
<tr>
<td>sin⁻¹( ), arcsine</td>
<td>878</td>
</tr>
<tr>
<td>sinh( ), hyperbolic sine</td>
<td>878</td>
</tr>
<tr>
<td>sinh⁻¹( ), hyperbolic arcsine</td>
<td>878</td>
</tr>
<tr>
<td>SinReg, sinusoidal regression</td>
<td>548, 879, 946</td>
</tr>
<tr>
<td>sinusoidal regression, SinReg</td>
<td>548, 879, 946</td>
</tr>
<tr>
<td>slope field, SLPFLD</td>
<td>415, 423, 448</td>
</tr>
<tr>
<td>SLPFLD, slope field</td>
<td>415, 423, 448</td>
</tr>
<tr>
<td>Smart Graph</td>
<td>318</td>
</tr>
<tr>
<td>SocialSt (social studies) category</td>
<td>31</td>
</tr>
<tr>
<td>software version</td>
<td>233</td>
</tr>
<tr>
<td>Solution Method graph format</td>
<td>414</td>
</tr>
<tr>
<td>solution, deSolve( )</td>
<td>266, 444, 809</td>
</tr>
<tr>
<td>solve( ), solve</td>
<td>77, 83, 446, 879</td>
</tr>
<tr>
<td>solve, solve( )</td>
<td>77, 83, 240, 242, 243, 247, 253, 256, 260, 446</td>
</tr>
<tr>
<td>solving linear equations</td>
<td>77, 260</td>
</tr>
<tr>
<td>SortA, sort ascending</td>
<td>882</td>
</tr>
<tr>
<td>SortD, sort descending</td>
<td>882</td>
</tr>
<tr>
<td>special characters</td>
<td>655, 656, 658</td>
</tr>
<tr>
<td>spherical vector display, (Sphere</td>
<td>882</td>
</tr>
<tr>
<td>Split App mode</td>
<td>188, 925</td>
</tr>
<tr>
<td>split screen</td>
<td>482, 485, 663, 676, 678</td>
</tr>
<tr>
<td>entry line</td>
<td>512, 514</td>
</tr>
<tr>
<td>exiting</td>
<td>509</td>
</tr>
<tr>
<td>setting</td>
<td>506</td>
</tr>
<tr>
<td>switch, switch( )</td>
<td>617, 886</td>
</tr>
<tr>
<td>switching</td>
<td>512</td>
</tr>
<tr>
<td>Split Screen mode</td>
<td>188, 924</td>
</tr>
<tr>
<td>split-screen mode</td>
<td></td>
</tr>
<tr>
<td>active graph</td>
<td>38</td>
</tr>
<tr>
<td>exiting</td>
<td>64</td>
</tr>
<tr>
<td>Number of graphs</td>
<td>62</td>
</tr>
<tr>
<td>returning from within an App</td>
<td>31</td>
</tr>
<tr>
<td>selecting active App</td>
<td>64</td>
</tr>
<tr>
<td>setting</td>
<td>59</td>
</tr>
<tr>
<td>setting initial Apps</td>
<td>61</td>
</tr>
<tr>
<td>specifying Apps displayed</td>
<td>62</td>
</tr>
<tr>
<td>Split 1 App</td>
<td>62</td>
</tr>
<tr>
<td>Split 2 App</td>
<td>62</td>
</tr>
<tr>
<td>status</td>
<td>36</td>
</tr>
<tr>
<td>status and open Apps</td>
<td>6</td>
</tr>
<tr>
<td>viewing</td>
<td>18</td>
</tr>
<tr>
<td>square root, $\sqrt{}$</td>
<td>909</td>
</tr>
<tr>
<td>standard annuity activity</td>
<td>769</td>
</tr>
<tr>
<td>standard deviation, stdDev( )</td>
<td>883</td>
</tr>
<tr>
<td>start timer, startTmr( )</td>
<td>882</td>
</tr>
<tr>
<td>startTmr( ), start timer</td>
<td>882</td>
</tr>
<tr>
<td>statistics</td>
<td>540–571</td>
</tr>
<tr>
<td>Box Plot</td>
<td>556</td>
</tr>
<tr>
<td>Calculation Type</td>
<td>541, 546</td>
</tr>
<tr>
<td>categories</td>
<td>565</td>
</tr>
<tr>
<td>Category</td>
<td>541, 543</td>
</tr>
<tr>
<td>combinations, nCr( )</td>
<td>846</td>
</tr>
<tr>
<td>factorial, !</td>
<td>74, 907</td>
</tr>
<tr>
<td>Freq</td>
<td>541, 543</td>
</tr>
<tr>
<td>frequency</td>
<td>563</td>
</tr>
<tr>
<td>Histogram plots</td>
<td>557</td>
</tr>
<tr>
<td>mean, mean( )</td>
<td>843</td>
</tr>
<tr>
<td>median, median( )</td>
<td>843</td>
</tr>
<tr>
<td>new plot, NewPlot</td>
<td>556, 848</td>
</tr>
<tr>
<td>operations</td>
<td>785</td>
</tr>
<tr>
<td>overview</td>
<td>540</td>
</tr>
<tr>
<td>permutations, nPr( )</td>
<td>849</td>
</tr>
<tr>
<td>plots</td>
<td>551, 553, 554, 558, 560</td>
</tr>
<tr>
<td>plots off, PlotsOff</td>
<td>311, 628, 855</td>
</tr>
<tr>
<td>plots on, PlotsOn</td>
<td>311, 628, 855</td>
</tr>
<tr>
<td>random norm, randNorm( )</td>
<td>863</td>
</tr>
<tr>
<td>random number seed, RandSeed</td>
<td>747, 863</td>
</tr>
<tr>
<td>scattering plots</td>
<td>555</td>
</tr>
<tr>
<td>show results, ShowStat</td>
<td>548, 876</td>
</tr>
<tr>
<td>standard deviation, stdDev( )</td>
<td>883</td>
</tr>
<tr>
<td>two-variable results, TwoVar</td>
<td>546, 893</td>
</tr>
<tr>
<td>variables</td>
<td>548</td>
</tr>
<tr>
<td>variance, variance( )</td>
<td>894</td>
</tr>
<tr>
<td>xyline plots</td>
<td>555</td>
</tr>
<tr>
<td>statistics. See also regressions</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td></td>
</tr>
<tr>
<td>battery low</td>
<td>71</td>
</tr>
<tr>
<td>on Apps desktop</td>
<td>6</td>
</tr>
<tr>
<td>split-screen</td>
<td>36</td>
</tr>
<tr>
<td>status line</td>
<td>201, 305</td>
</tr>
<tr>
<td>command parameters</td>
<td>23</td>
</tr>
<tr>
<td>history information</td>
<td>27</td>
</tr>
<tr>
<td>stdDev( ), standard deviation</td>
<td>883</td>
</tr>
<tr>
<td>stdDevPop</td>
<td>883</td>
</tr>
<tr>
<td>StoGDB, store graph database</td>
<td>505, 629, 884</td>
</tr>
<tr>
<td>Stop, stop</td>
<td>884</td>
</tr>
<tr>
<td>stop, Stop</td>
<td>579</td>
</tr>
<tr>
<td>StoPic, store picture</td>
<td>629, 884</td>
</tr>
<tr>
<td>stopping a calculation</td>
<td>165</td>
</tr>
<tr>
<td>Store (STO) key</td>
<td>17</td>
</tr>
<tr>
<td>storing</td>
<td></td>
</tr>
<tr>
<td>graph database, StoGDB</td>
<td>629, 884</td>
</tr>
<tr>
<td>picture, StoPic</td>
<td>629, 884</td>
</tr>
<tr>
<td>symbol, $\Rightarrow$</td>
<td>596, 913</td>
</tr>
<tr>
<td>string( ), expression to string</td>
<td>603, 884</td>
</tr>
<tr>
<td>strings</td>
<td></td>
</tr>
<tr>
<td>append, &amp;</td>
<td>602, 908</td>
</tr>
</tbody>
</table>
character code, ord( ) .......... 602
character string, char( ) .... 602, 792
dimension, dim( ) .............. 602
expression to string, string( ) 603, 884
format, format( ) .... 602, 620, 627, 823
indirection, # ............ 602, 909, 944
inputting, InputSt .... 601, 619, 728
left, left( ) ............. 602, 835
mid-string, mid( ) .......... 602, 844
operations ............ 600, 602, 785
right, right( ) .......... 603, 866
rotate, rotate( ) .......... 603, 866
shift, shift( ) .......... 603, 875
string to expression, expr( ) 601, 602, .......... 620, 819
within, InString .......... 602, 833
Style, style ................. 885
style, Style ................. 313, 628
subMat( ), submatrix ........ 885
submenus ................. 177
substitutions ............ 249, 251, 253
subtract, – ............... 902
subtraction key (□) ........ 14
sum( ), summation .......... 866, 885
sum, Σ( ) .................. 266, 909
support and service .......... 965
switch( ), switch ........... 886
switch, switch( ) .......... 617
symbolic manipulation .......... 269
system data, sysData .......... 469, 470
system variables .......... 942, 943

T
T, transpose ................. 886
t0 window variable .......... 416
TABLE SETUP, table setup .......... 455
Table, build table .......... 887
table-graph, Graph<->Table .......... 456
tables
Δtbl ......................... 456
automatic ................. 459
cell width ............... 461, 467
clearing, ClrTable ............ 794
complex numbers .......... 463
differential equations .......... 452
displaying, DispTbl .......... 620, 627, 812
functions ................. 463
generating with sequence .......... 373
graphing, Graph<->Table .......... 456
incrementing, Δtbl .......... 456
Independent AUTO/ASK .. 457, 459, 464
manual ......................... 464
overview ......................... 454
programs ......................... 627
setTable( ) .................. 459
setting, setTable( ) .......... 617, 873
setup ......................... 459
setup, TABLE SETUP .......... 455
starting, tblStart .......... 456
tblStart ......................... 456
tan( ), tangent .......................... 887
\( \tan^{-1}( ) \), arctangent ............... 888
Tangent (graph math tool) 331, 336, 346, 353
tangent, \( \tan( ) \) .......................... 887
tanh( ), hyperbolic tangent ............... 888
tanh\(^{-1}( ) \), hyperbolic arctangent .... 889
Taylor polynomial, taylor( ) 266, 269, 889
taylor( ), Taylor polynomial ............... 889
tblStart, table start ....................... 456
tCollect( ), trigonometric collection . 889
temperature conversion, tmpCnv( ) .286, 891
temperature-range conversion,
\( \Delta \text{tmpCnv}( ) \) .......................... 286
temperature-range conversion,
\( \Delta \text{tmpCnv}( ) \) .......................... 891
tExpand( ), trigonometric expansion . 890
text editing .............................. 645, 655, 661
cut, copy, paste ......................... 212, 653
find ........................................ 654
highlighting .............................. 652
Text, text .................................. 890
text, Text .................................. 621, 622
Then, Then ............................... 606, 607, 831
three-dimensional graphing 375–409
animation ................................. 101, 391
CONTOUR LEVELS ......................... 104, 395
HIDDEN SURFACE ........................ 104, 395
WIRE AND CONTOUR ..................... 104, 395
WIRE FRAME ............................... 104, 395

TI Connectivity Cable 65, 716, 733, 737
TI ViewScreen overhead panel
connecting ................................. 68
TI Connect software ...................... 65, 733
time
reset ....................................... 48
setting ..................................... 40
time plots, TIME 357, 363, 434, 435
time value of money activity ............. 771
TIME, time plots 357, 434, 435
timeCnv( ), convert time ................. 890
TI-Presenter video adapter
connecting ................................. 68
Title, title .................................. 890
tmax window variable .................... 351, 416
tmin window variable .................... 351
tmpCnv( ), temperature conversion . 286,
891
toolbar
define, Custom 621, 803
off, CustmOff 230, 803
on, CustmOn 230, 803
Toolbar menus
calculator Home screen 49
moving among 55
replaced by custom menu 56
selecting math operations 14
Toolbar, toolbar 621, 892
top-bottom split screen
setting 59
setting initial Apps 61
while, While .................................. 613
Window Editor ................................. 58
window variables
  (x ........................................... 941
  (y ........................................... 941
diftol ....................................... 418
dtime ........................................ 418
Estep ......................................... 418
eyeϕ (z axis) ................................. 378, 387, 389
eyeθ (x axis) ................................. 378, 387, 389
eyeψ (rotation) ............................. 378, 387, 389
fldres ....................................... 418
ncontour ..................................... 379
ncurves ....................................... 417
nmax .......................................... 359
nmin .......................................... 359
plotStep ..................................... 359
plotStrt ..................................... 359
θmax .......................................... 343
θmin .......................................... 343
θstep ......................................... 344
t0 ............................................. 416
tmax .......................................... 351, 416
tplot .......................................... 417
tstep .......................................... 351, 416
xgrid ......................................... 379
xmax ......................................... 313, 344, 351, 359, 379, 417, 941
xmin ......................................... 313, 344, 351, 359, 379, 417, 941
xres .......................................... 314
xscl ........................................... 314, 344, 352, 359, 417
ygrid .......................................... 379
ymax ........................................... 313, 344, 351, 359, 379, 417, 941
ymin ........................................... 313, 344, 351, 359, 379, 417, 941
yscl ........................................... 314, 344, 352, 359, 417
zmax .......................................... 379
zmin .......................................... 379
wire-and-contour graphing .............. 104, 395
wire-frame graphing ....................... 104, 395
with, | ............................. 80, 83, 239, 249, 913, 944
within string, inString( ) .............. 602, 833

X
xgrid window variable ...................... 379
xmax window variable ........................ 313, 344, 351, 359, 379, 417, 941
xmin window variable 313, 344, 351, 359, .................................. 379, 417, 941
xor, Boolean exclusive or ................. 605, 685, 895
XorPic, exclusive or picture ............... 629, 896
xres window variable ........................ 314
xscl window variable 314, 344, 352, 359, 417, .................................. 941
xyline plots .................................. 555

Y
Y= editor 86, 89, 305, 343, 349, 356, 377, .................................. 412, 471
ygrid window variable ...................... 379
ymax window variable ... 313, 344, 359, 379, ................. 417, 941
ymin window variable 313, 344, 351, 359, ................. 379, 417, 941
yscl window variable 314, 344, 352, 359, .... 417

Z
Zero (graph math tool) ........... 331, 333
zeroes
activity ................................ 766
zeroes, zeroes( ) ........... 240, 256, 263
zeroes, zeros( ) ............... 743, 896
zeros( ), zeroes .............. 743, 896
zmax window variable .......... 379
zmin window variable .......... 379
zoom
box, ZoomBox ........ 325, 327, 897
data, ZoomData .......... 326, 898
decimal, ZoomDec ........ 325, 898
factors ................. 326, 328
fit, ZoomFit ............ 326, 899
in, ZoomIn ............ 325, 328, 899
integer, ZoomInt ....... 326, 899
Memory .................. 326, 329
out, ZoomOut .......... 325, 328, 900
previous, ZoomPrev .......... 330, 900
recall, ZoomRcl ......... 330, 900
square, ZoomSqr ....... 325, 900
standard, ZoomStd ...... 326, 901
store, ZoomSto ........ 329, 330, 901

trig, ZoomTrig .......... 326, 901
Zoom menu .................. 325
ZoomBox, zoom box ........ 897
ZoomData, zoom data ....... 898
ZoomDec, zoom decimal ....... 898
ZoomFit, zoom fit .......... 899
ZoomIn, zoom in ........... 899
ZoomInt, zoom integer ...... 899
ZoomOut, zoom out .......... 900
ZoomPrev, zoom previous ... 900
ZoomRcl, zoom recall ...... 900
ZoomSqr, zoom square ...... 900
ZoomStd, zoom standard ... 901
ZoomSto, zoom store ...... 901
ZoomTrig, zoom trig ...... 901